Manuale PHP

Stig Sæther Bakken

Alexander Aulbach

Egon Schmid

Jim Winstead

Lars Torben Wilson

Rasmus Lerdorf

Andrei Zmievski

Jouni Ahto

A cura di

Luca Perugini

Simone Cortesi

Tradotto con la collaborazione di:
Marco Cucinato
Massimo Colombo
Marco De Nittis
Darvin Andrioli
Fabio Gandola
Sergio Marchesini
Alan D’Angelo
Giacomo Tesio
Marco Spisto
Gabriele Scaroni
Mariano Calandra
Rocco Curcio
Luca Costantino
21-05-2002

Manuale PHP

Stig Sæther Bakken, Alexander Aulbach, Egon Schmid, Jim Winstead, Lars Torben Wilson, Rasmus Lerdorf, Andrei Zmievski, e Jouni Ahto

A cura di Luca Perugini

A cura di Simone Cortesi

Tradotto con la collaborazione di:
Marco Cucinato
Massimo Colombo
Marco De Nittis
Darvin Andrioli
Fabio Gandola
Sergio Marchesini
Alan D'Angelo
Giacomo Tesio
Marco Spisto
Gabriele Scaroni
Mariano Calandra
Rocco Curcio
Luca Costantino

Pubblicato 21-05-2002

Sommario

Prefazione

PHP, che significa "PHP: Hypertext Preprocessor", è un linguaggio di scripting general-purpose Open Source molto utilizzato, è specialmente indicato per lo sviluppo Web e può essere integrato nell'HTML. La sua sintassi è basata su quella di C, Java e Perl, ed è molto semplice da imparare. L'obiettivo principale del linguaggio è quello di permettere agli sviluppatori web di scrivere velocemente pagine web dinamiche, ma con PHP si possono fare molte altre cose.

Questo manuale consiste principalmente in un elenco commentato di funzioni ma contiene anche una guida al linguaggio, spiegazione delle maggiori caratteristiche del PHP e altre informazioni aggiuntive.

Il manuale è fornito in diversi formati qui: http://www.php.net/docs.php. I file sono aggiornati ogni volta che i contenuti cambiano. Maggiori informazioni su come questo manuale viene sviluppato possono essere trovate nell'appendice 'Informazioni sul Manuale'.

Parte I. Guida Rapida

Capitolo 1. Introduzione

Che cos'è il PHP?

PHP (acronimo ricorsivo per "PHP: Hypertext Preprocessor") è un linguaggio di scripting general-purpose Open Source molto utilizzato, è specialmente indicato per lo sviluppo Web e può essere integrato nell'HTML.

Risposta banale, ma che cosa significa? Un esempio:

Esempio 1-1. Un esempio introduttivo

```
<html>
    <head>
        <title>Esempio</title>
    </head>
    <body>

        <?php
        echo "Ciao, sono uno script PHP!";
        ?>

    </body>
</html>
```

Notate come questo esempio è differente da uno script scritto in altri linguaggi tipo Perl o C -- invece di scrivere un programma con parecchi comandi per produrre HTML, si scrive in HTML con qualche comando immerso per ottenere dei risultati (in questo semplice esempio, la visualizzazione di una frase). Il codice PHP è delimitato da speciali start ed end tag che ne indicano l'inizio e la fine e che consentono di passare dal modo HTML al modo PHP.

What distinguishes PHP from something like client-side JavaScript is that the code is executed on the server. If you were to have a script similar to the above on your server, the client would receive the results of running that script, with no way of determining what the underlying code may be. You can even configure your web server to process all your HTML files with PHP, and then there's really no way that users can tell what you have up your sleeve.

The best things in using PHP are that it is extremely simple for a newcomer, but offers many advanced features for a professional programmer. Don't be afraid reading the long list of PHP's features. You can jump in, in a short time, and start writing simple scripts in a few hours.

Although PHP's development is focused on server-side scripting, you can do much more with it. Read on, and see more in the What can PHP do? section.

What can PHP do?

Anything. PHP is mainly focused on server-side scripting, so you can do anything any other CGI program can do, such as collect form data, generate dynamic page content, or send and receive cookies. But PHP can do much more.

There are three main fields where PHP scripts are used.

- Server-side scripting. This is the most traditional and main target field for PHP. You need three things to make this work. The PHP parser (CGI or server module), a webserver and a web browser. You need to run the webserver, with a connected PHP installation. You can access the PHP program output with a web browser, viewing the PHP page through the server. See the installation instructions section for more information.
- Command line scripting. You can make a PHP script to run it without any server and any browser. You only need the PHP parser to use it this way. This type of usage is ideal for scripts regularly executed using cron (task sheduler on Windows), or simple text processing tasks. See the section about Command line usage of PHP for more information.
- Writing client-side GUI applications. PHP is probably not the very best language to write windowing applications, but if you know PHP very well, and would like to use some advanced PHP features in your client-side applications you can also use PHP-GTK to write such programs. You also have the ability to write cross-platform applications this way. PHP-GTK is an extension to PHP, not available in the main distribution. If you are interested in PHP-GTK, visit it's own website (http://gtk.php.net/).

PHP can be used on all major operating systems, including Linux, many Unix variants (including HP-UX, Solaris and OpenBSD), Microsoft Windows, Mac OS X, RISC OS, and probably others. PHP has also support for most of the web servers today. This includes Apache, Microsoft Internet Information Server, Personal Web Server, Netscape and iPlanet servers, Oreilly Website Pro server, Caudium, Xitami, OmniHTTPd, and many others. For the majority of the servers PHP has a module, for the others supporting the CGI standard, PHP can work as a CGI processor.

So with PHP, you have the freedom of choosing an operating system and a web server. Furthermore, you also have the choice of using procedural programming or object oriented programming, or a mixture of them. Although not every standard OOP feature is realized in the current version of PHP, many code libraries and large applications (including the PEAR library) are written only using OOP code.

With PHP you are not limited to output HTML. PHP's abilities includes outputing images, PDF files and even Flash movies (using libswf and Ming) generated on the fly. You can also output easily any text, such as XHTML and any other XML file. PHP can autogenerate these files, and save them in the file system, instead of printing it out, forming a server-side cache for your dynamic content.

One of the strongest and most significant feature in PHP is its support for a wide range of databases. Writing a database-enabled web page is incredibly simple. The following databases are currently supported:

Adabas D	Ingres	Oracle (OCI7 and OCI8)
dBase	InterBase	Ovrimos
Empress	FrontBase	PostgreSQL
FilePro (read-only)	mSQL	Solid
Hyperwave	Direct MS-SQL	Sybase
IBM DB2	MySQL	Velocis
Informix	ODBC	Unix dbm

We also have a DBX database abstraction extension allowing you to transparently use any database supported by that extension. Additionally PHP supports ODBC, the Open Database Connection standard, so you can connect to any other database supporting this world standard.

PHP also has support for talking to other services using protocols such as LDAP, IMAP, SNMP, NNTP, POP3, HTTP, COM (on Windows) and countless others. You can also open raw network sockets and interact using any other protocol. PHP has support for the WDDX complex data exchange between virtually all Web programming languages. Talking about interconnection, PHP has support for instantiation of Java objects and using them transparently as PHP objects. You can also use our CORBA extension to access remote objects.

PHP has extremely useful text processing features, from the POSIX Extended or Perl regular expressions to parsing XML documents. For parsing and accessing XML documents, we support the SAX and DOM standards. You can use our XSLT extension to transform XML documents.

While using PHP in the ecommerce field, you'll find the Cybercash payment, CyberMUT, Verysign Payflow Pro and CCVS functions useful for your online payment programs.

At last but not least, we have many other interesting extensions, the mnoGoSearch search engine functions, the IRC Gateway functions, many compression utilities (gzip, bz2), calendar conversion, translation...

As you can see this page is not enough to list all the features and benefits PHP can offer. Read on in the sections about installing PHP, and see the function reference part for explanation of the extensions mentioned here.

A brief history of PHP

PHP was conceived sometime in the fall of 1994 by Rasmus Lerdorf (mailto:rasmus@php.net). Early non-released versions were used on his home page to keep track of who was looking at his online resume. The first version used by others was available sometime in early 1995 and was known as the Personal Home Page Tools. It consisted of a very simplistic parser engine that only understood a few special macros and a number of utilities that were in common use on home pages back then. A guestbook, a counter and some other stuff. The parser was rewritten in mid-1995 and named PHP/FI Version 2. The FI came from another package Rasmus had written which interpreted html form data. He combined the Personal Home Page tools scripts with the Form Interpreter and added mSQL support and PHP/FI was born. PHP/FI grew at an amazing pace and people started contributing code to it.

It is difficult to give any hard statistics, but it is estimated that by late 1996 PHP/FI was in use on at least 15,000 web sites around the world. By mid-1997 this number had grown to over 50,000. Mid-1997 also saw a change in the development of PHP. It changed from being Rasmus' own pet project that a handful of people had contributed to, to being a much more organized team effort. The parser was rewritten from scratch by Zeev Suraski and Andi Gutmans and this new parser formed the basis for PHP Version 3. A lot of the utility code from PHP/FI was ported over to PHP 3 and a lot of it was completely rewritten.

The latest version (PHP 4) uses the Zend (http://www.zend.com/) scripting engine to deliver higher performance, supports an even wider array of third-party libraries and extensions, and runs as a native server module with all of the popular web servers.

Today (1/2001) PHP 3 or PHP 4 now ships with a number of commercial products such as Red Hat's Stronghold web server. A conservative estimate based on an extrapolation from numbers provided by Netcraft (http://www.netcraft.com/) (see also Netcraft Web Server Survey (http://www.netcraft.com/survey/)) would be that PHP is in use on over 5,100,000 sites around the world. To put that in perspective, that is slightly more sites than run Microsoft's IIS server on the Internet (5.03 million).

Capitolo 2. Installazione

General Installation Considerations

Before installing first, you need to know what do you want to use PHP for. There are three main fields you can use PHP, as described in the What can PHP do? section:

- Server-side scripting
- Command line scripting
- Client-side GUI applications

For the first and most common form, you need three things: PHP itself, a web server and a web browser. You probably already have a web browser, and depending on your operating system setup, you may also have a web server (eg. Apache on Linux or IIS on Windows). You may also rent webspace at a company. This way, you don't need to set up anything on your own, only write your PHP scripts, upload it to the server you rent, and see the results in your browser. You can find a list of hosting companies at http://hosts.php.net/.

While setting up the server and PHP on your own, you have two choices for the method of connecting PHP to the server. For many servers PHP has a direct module interface (also called SAPI). These servers include Apache, Microsoft Internet Information Server, Netscape and iPlanet servers. Many other servers have support for ISAPI, the Microsoft module interface (OmniHTTPd for example). If PHP has no module support for your web server, you can always use it as a CGI processor. This means you set up your server to use the command line executable of PHP (`php.exe` on Windows) to process all PHP file requests on the server.

If you are also interested to use PHP for command line scripting (eg. write scripts autogenerating some images for you offline, or processing text files depending on some arguments you pass to them), you always need the command line executable. For more information, read the section about writing command line PHP applications. In this case, you need no server and no browser.

With PHP you can also write client side GUI applications using the PHP-GTK extension. This is a completely different approach than writing web pages, as you do not output any HTML, but manage windows and objects within them. For more information about PHP-GTK, please visit the site dedicated to this extension (http://gtk.php.net/). PHP-GTK is not included in the official PHP distribution.

From now on, this section deals with setting up PHP for web servers on Unix and Windows with server module interfaces and CGI executables.

Downloading PHP, the source code, and binary distributions for Windows can be found at http://www.php.net/. We recommend you to choose a mirror (http://www.php.net/mirrors.php) nearest to you for downloading the distributions.

Unix/HP-UX installs

This section contains notes and hints specific to installing PHP on HP-UX systems.

Esempio 2-1. Installation Instructions for HP-UX 10

```
From: paul_mckay@clearwater-it.co.uk
04-Jan-2001 09:49
```

```
(These tips are for PHP 4.0.4 and Apache v1.3.9)

So you want to install PHP and Apache on a HP-UX 10.20 box?

1. You need gzip, download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/gzip-1.2.4a/gzip-1.2.4a-sd-10.20.depot.Z
uncompress the file and install using swinstall

2. You need gcc, download a binary distribution from
http://gatekeep.cs.utah.edu/ftp/hpux/Gnu/gcc-2.95.2/gcc-2.95.2-sd-10.20.depot.gz
gunzip this file and install gcc using swinstall.

3. You need the GNU binutils, you can download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/binutils-2.9.1/binutils-2.9.1-sd-10.20.depot.gz
gunzip and install using swinstall.

4. You now need bison, you can download a binary distribution from
http://hpux.connect.org.uk/ftp/hpux/Gnu/bison-1.28/bison-1.28-sd-10.20.depot.gz
install as above.

5. You now need flex, you need to download the source from one of the
http://www.gnu.org mirrors. It is in the <filename>non-gnu</filename> direc-
tory of the ftp site.
Download the file, gunzip, then tar -xvf it. Go into the newly created flex
directory and do a ./configure, then a make, and then a make install

If you have errors here, it’s probably because gcc etc. are not in your
PATH so add them to your PATH.

Right, now into the hard stuff.

6. Download the PHP and apache sources.

7. gunzip and tar -xvf them.

We need to hack a couple of files so that they can compile ok.

8. Firstly the configure file needs to be hacked because it seems to lose
track of the fact that you are a hpux machine, there will be a
better way of doing this but a cheap and cheerful hack is to put
    lt_target=hpux10.20
on line 47286 of the configure script.

9. Next, the Apache GuessOS file needs to be hacked. Under
apache_1.3.9/src/helpers change line 89 from
    "echo "hp${HPUXMACH}-hpux${HPUXVER}"; exit 0"
to:
    "echo "hp${HPUXMACH}-hp-hpux${HPUXVER}"; exit 0"

10. You cannot install PHP as a shared object under HP-UX so you must compile
it as a static, just follow the instructions at the Apache page.

11. PHP and apache should have compiled OK, but Apache won’t start. you need
to create a new user for Apache, eg www, or apache. You then change lines 252
and 253 of the conf/httpd.conf in Apache so that instead of
    User nobody
    Group nogroup
```

```
you have something like
    User www
    Group sys

This is because you can’t run Apache as nobody under hp-ux.
Apache and PHP should then work.

Hope this helps somebody,
Paul Mckay.
```

Unix/Linux installs

This section contains notes and hints specific to installing PHP on Linux distributions.

Using Packages

Many Linux distributions have some sort of package installation system, such as RPM. This can assist in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build PHP and/or your webserver. If you are unfamiliar with building and compiling your own software, it is worth checking to see whether somebody has already built a packaged version of PHP with the features you need.

Unix/Mac OS X installs

This section contains notes and hints specific to installing PHP on Mac OS X Server.

Using Packages

There are a few pre-packaged and pre-compiled versions of PHP for Mac OS X. This can help in setting up a standard configuration, but if you need to have a different set of features (such as a secure server, or a different database driver), you may need to build PHP and/or your web server yourself. If you are unfamiliar with building and compiling your own software, it’s worth checking whether somebody has already built a packaged version of PHP with the features you need.

Compiling for OS X server

There are two slightly different versions of Mac OS X, client and server. The following is for OS X Server.

Esempio 2-2. Mac OS X server install

```
1. Get the latest distributions of Apache and PHP
2. Untar them, and run the configure program on Apache like so.
    ./configure --exec-prefix=/usr \
```

```
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache

4. You may also want to add this line:
    setenv OPTIM=-O2
    If you want the compiler to do some optimization.

5. Next, go to the PHP 4 source directory and configure it.
    ./configure --prefix=/usr \
    --sysconfdir=/etc \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --with-xml \
    --with-apache=/src/apache_1.3.12

    If you have any other additions (MySQL, GD, etc.), be sure to add
    them here. For the --with-apache string, put in the path to your
    apache source directory, for example "/src/apache_1.3.12".
6. make
7. make install
    This will add a directory to your Apache source directory under
    src/modules/php4.

8. Now, reconfigure Apache to build in PHP 4.
    ./configure --exec-prefix=/usr \
    --localstatedir=/var \
    --mandir=/usr/share/man \
    --libexecdir=/System/Library/Apache/Modules \
    --iconsdir=/System/Library/Apache/Icons \
    --includedir=/System/Library/Frameworks/Apache.framework/Versions/1.3/Headers \
    --enable-shared=max \
    --enable-module=most \
    --target=apache \
    --activate-module=src/modules/php4/libphp4.a

    You may get a message telling you that libmodphp4.a is out of date.
    If so, go to the src/modules/php4 directory inside your apache
    source directory and run this command:

    ranlib libmodphp4.a

    Then go back to the root of the apache source directory and run the
    above configure command again. That’ll bring the link table up to
    date.

9. make

10. make install

11. copy and rename the php.ini-dist file to your "bin" directory from your
    PHP 4 source directory:
```

```
cp php.ini-dist /usr/local/bin/php.ini

or (if your don't have a local directory)

cp php.ini-dist /usr/bin/php.ini
```

Other examples for Mac OS X client (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) and Mac OS X server (http://www.stepwise.com/Articles/Workbench/Apache-1.3.14-MacOSX.html) are available at Stepwise (http://www.stepwise.com/).

Compiling for MacOS X client

Those tips are graciously provided by Marc Liyanage (http://www.entropy.ch/software/macosx).

The PHP module for the Apache web server included in Mac OS X. This version includes support for the MySQL and PostgreSQL databases.

NOTE: Be careful when you do this, you could screw up your Apache web server!

Do this to install:

- 1. Open a terminal window
- 2. Type "wget http://www.diax.ch/users/liyanage/software/macosx/libphp4.so.gz", wait for download to finish
- 3. Type "gunzip libphp4.so.gz"
- 4. Type "sudo apxs -i -a -n php4 libphp4.so"

Now type "`sudo open -a TextEdit /etc/httpd/httpd.conf`" TextEdit will open with the web server configuration file. Locate these two lines towards the end of the file: (Use the Find command)

```
#AddType application/x-httpd-php .php
#AddType application/x-httpd-php-source .phps
```

Remove the two hash marks (#), then save the file and quit TextEdit.

Finally, type "`sudo apachectl graceful`" to restart the web server.

PHP should now be up and running. You can test it by dropping a file into your "Sites" folder which is called "test.php". Into that file, write this line: "`<?php phpinfo() ?>`".

Now open up `127.0.0.1/~your_username/test.php` in your web browser. You should see a status table with information about the PHP module.

Unix/OpenBSD installs

This section contains notes and hints specific to installing PHP on OpenBSD (http://www.openbsd.org/).

Using Ports

This is the recommended method of installing PHP on OpenBSD, as it will have the latest patches and security fixes applied to it by the maintainers. To use this method, ensure that you have a recent ports tree (http://www.openbsd.org/ports.html). Then simply find out which flavors you wish to install, and issue the **make install** command. Below is an example of how to do this.

Esempio 2-3. OpenBSD Ports Install Example

```
$ cd /usr/ports/www/php4
$ make show VARNAME=FLAVORS
 (choose which flavors you want from the list)
$ env FLAVOR="imap gettext ldap mysql gd" make install
$ /usr/local/sbin/php4-enable
```

Using Packages

There are pre-compiled packages available for your release of OpenBSD (http://www.openbsd.org/). These integrate automatically with the version of Apache installed with the OS. However, since there are a large number of options (called *flavors*) available for this port, you may find it easier to compile it from source using the ports tree. Read the packages(7) (http://www.openbsd.org/cgi-bin/man.cgi?query=packages) manual page for more information in what packages are available.

Unix/Solaris installs

This section contains notes and hints specific to installing PHP on Solaris systems.

Required software

Solaris installs often lack C compilers and their related tools. The required software is as follows:

- gcc (recommended, other C compilers may work)
- make
- flex
- bison
- m4

- autoconf
- automake
- perl
- gzip
- tar

In addition, you will need to install (and possibly compile) any additional software specific to your configuration, such as Oracle or MySQL.

Using Packages

You can simplify the Solaris install process by using pkgadd to install most of your needed components.

Installation on UNIX systems

This section will guide you through the general configuration and installation of PHP on Unix systems. Be sure to investigate any sections specific to your platform or web server before you begin the process.

Prerequisite knowledge and software:

- Basic UNIX skills (being able to operate "make" and a C compiler, if compiling)
- An ANSI C compiler (if compiling)
- flex (for compiling)
- bison (for compiling)
- A web server
- Any module specific components (such as gd, pdf libs, etc.)

There are several ways to install PHP for the Unix platform, either with a compile and configure process, or through various pre-packaged methods. This documentation is mainly focused around the process of compiling and configuring PHP.

The initial PHP setup and configuration process is controlled by the use of the commandline options of the `configure` script. This page outlines the usage of the most common options, but there are many others to play with. Check out the Complete list of configure options for an exhaustive rundown. There are several ways to install PHP:

- As an Apache module
- As an fhttpd module
- For use with AOLServer, NSAPI, phttpd, Pi3Web, Roxen, thttpd, or Zeus.
- As a CGI executable

Apache Module Quick Reference

PHP can be compiled in a number of different ways, but one of the most popular is as an Apache module. The following is a quick installation overview.

Esempio 2-4. Quick Installation Instructions for PHP 4 (Apache Module Version)

```
1.  gunzip apache_1.3.x.tar.gz
2.  tar xvf apache_1.3.x.tar
3.  gunzip php-x.x.x.tar.gz
4.  tar xvf php-x.x.x.tar
5.  cd apache_1.3.x
6.  ./configure --prefix=/www
7.  cd ../php-x.x.x
8.  ./configure --with-mysql --with-apache=../apache_1.3.x --enable-track-vars
9.  make
10. make install
11. cd ../apache_1.3.x
12. ./configure --activate-module=src/modules/php4/libphp4.a
13. make
14. make install
15. cd ../php-x.x.x
16. cp php.ini-dist /usr/local/lib/php.ini
17. Edit your httpd.conf or srm.conf file and add:
      AddType application/x-httpd-php .php

18. Use your normal procedure for restarting the Apache server. (You must
    stop and restart the server, not just cause the server to reload by
    use a HUP or USR1 signal.)
```

Building

When PHP is configured, you are ready to build the CGI executable. The command **make** should take care of this. If it fails and you can’t figure out why, see the Problems section.

Installation on Windows systems

This section applies to Windows 95/98/Me and Windows NT/2000/XP. Do not expect PHP to work on 16 bit platforms such as Windows 3.1. Sometimes we refer to the supported Windows platforms as Win32.

There are two main ways to install PHP for Windows: either manually or by using the InstallShield installer.

If you have Microsoft Visual Studio, you can also build PHP from the original source code.

Once you have PHP installed on your Windows system, you may also want to load various extensions for added functionality.

Windows InstallShield

The Windows PHP installer available from the downloads page at http://www.php.net/, this installs the CGI version of PHP and, for IIS, PWS, and Xitami, configures the web server as well. Also note, that while the InstallShield installer is an easy way to make PHP work, it is restricted in many aspects, as automatic setup of extensions for example is not supported.

Install your selected HTTP server on your system and make sure that it works.

Run the executable installer and follow the instructions provided by the installation wizard. Two types of installation are supported - standard, which provides sensible defaults for all the settings it can, and advanced, which asks questions as it goes along.

The installation wizard gathers enough information to set up the `php.ini` file and configure the web server to use PHP. For IIS and also PWS on NT Workstation, a list of all the nodes on the server with script map settings is displayed, and you can choose those nodes to which you wish to add the PHP script mappings.

Once the installation has completed the installer will inform you if you need to restart your system, restart the server, or just start using PHP.

Attenzione

Be aware, that this setup of PHP is not secure. If you would like to have a secure PHP setup, you'd better go on the manual way, and set every option carefully. This automatically working setup gives you an instantly working PHP installation, but it is not meant to be used on online servers.

Manual Installation Steps

This install guide will help you manually install and configure PHP on your Windows webserver. You need to download the zip binary distribution from the downloads page at http://www.php.net/. The original version of this guide was compiled by Bob Silva (mailto:bob_silva@mail.umesd.k12.or.us), and can be found at http://www.umesd.k12.or.us/php/win32install.html.

This guide provides manual installation support for:

- Personal Web Server 3 and 4 or newer
- Internet Information Server 3 and 4 or newer
- Apache 1.3.x
- OmniHTTPd 2.0b1 and up
- Oreilly Website Pro
- Xitami
- Netscape Enterprise Server, iPlanet

PHP 4 for Windows comes in two flavours - a CGI executable (php.exe), and several SAPI modules (for example: php4isapi.dll). The latter form is new to PHP 4, and provides significantly improved performance and some new functionality.

> **Attenzione**
>
> The SAPI modules have been significantly improved in the 4.1 release, however, you may find that you encounter possible server errors or other server modules such as ASP failing, in older systems.

If you choose one of the SAPI modules and use Windows 95, be sure to download the DCOM update from the Microsoft DCOM pages (http://download.microsoft.com/msdownload/dcom/95/x86/en/dcom95.exe). For the ISAPI module, an ISAPI 4.0 compliant Web server is required (tested on IIS 4.0, PWS 4.0 and IIS 5.0). IIS 3.0 is *NOT* supported. You should download and install the Windows NT 4.0 Option Pack with IIS 4.0 if you want native PHP support.

The following steps should be performed on all installations before the server specific instructions.

- Extract the distribution file to a directory of your choice. `c:\php\` is a good start. You probably do not want to use a path in which spaces are included (for example: c:\program files\php is not a good idea). Some web servers will crash if you do.
- You need to ensure that the DLLs which PHP uses can be found. The precise DLLs involved depend on which web server you use and whether you want to run PHP as a CGI or as a server module. `php4ts.dll` is always used. If you are using a server module (e.g. ISAPI or Apache) then you will need the relevant DLL from the `sapi` folder. If you are using any PHP extension DLLs then you will need those as well. To make sure that the DLLs can be found, you can either copy them to the system directory (e.g. `winnt/system32` or `windows/system`) or you can make sure that they live in the same directory as the main PHP executable or DLL your web server will use (e.g. `php.exe`, `php4apache.dll`).

  The PHP binary, the SAPI modules, and some extensions rely on external DLLs for execution. Make sure that these DLLs in the distribution exist in a directory that is in the Windows PATH. The best bet to do it is to copy the files below into your system directory, which is typically:
  `c:\windows\system` for Windows 9x/ME
  `c:\winnt\system32` for Windows NT/2000
  `c:\windows\system32` for Windows XP
  The files to copy are:
  `php4ts.dll`, if it already exists there, overwrite it
  The files in your distribution's 'dlls' directory. If you have them already installed on your system, overwrite them on

  Download the latest version of the Microsoft Data Access Components (MDAC) for your platform, especially if you use Microsoft Windows 9x/NT4. MDAC is available at http://www.microsoft.com/data/.

- Copy your chosen ini file (see below) to your '%WINDOWS%' directory on Windows 9x/Me or to your '%SYSTEMROOT%' directory under Windows NT/2000/XP and rename it to `php.ini`. Your '%WINDOWS%' or '%SYSTEMROOT%' directory is typically:
  `c:\windows` for Windows 9x/ME/XP
  `c:\winnt` or `c:\winnt40` for NT/2000 servers

  There are two ini files distributed in the zip file, `php.ini-dist` and `php.ini-optimized`. We advise you to use `php.ini-optimized`, because we optimized the default settings in this file for performance, and security. The best is to study all the ini settings and set every element manually yourself. If you would like to achieve the best security, then this is the way for you, although PHP works fine with these default ini files.

- Edit your new `php.ini` file:
  - You will need to change the 'extension_dir' setting to point to your php-install-dir, or where you have placed your `php_*.dll` files. ex: `c:\php\extensions`
  - If you are using OmniHTTPd, do not follow the next step. Set the 'doc_root' to point to your webservers document_root. For example: `c:\apache\htdocs` or `c:\webroot`
  - Choose which extensions you would like to load when PHP starts. See the section about Windows extensions, about how to set up one, and what is already built in. Note that on a new installation it is advisable to first get PHP working and tested without any extensions before enabling them in `php.ini`.
  - On PWS and IIS, you can set the `browscap.ini` to point to: `c:\windows\system\inetsrv\browscap.ini` on Windows 9x/Me, `c:\winnt\system32\inetsrv\browscap.ini` on NT/2000, and `c:\windows\system32\inetsrv\browscap.ini` on XP.
  - Note that the `mibs` directory supplied with the Windows distribution contains support files for SNMP. This directory should be moved to `DRIVE:\usr\mibs` (`DRIVE` being the drive where PHP is installed.)
  - If you're using NTFS on Windows NT, 2000 or XP, make sure that the user running the webserver has read permissions to your `php.ini` (e.g. make it readable by Everyone).
- For PWS give execution permission to the webroot:
  - Start PWS Web Manager
  - Edit Properties of the "Home"-Directory
  - Select the "execute"-Checkbox

Building from source

Before getting started, it is worthwhile answering the question: "Why is building on Windows so hard?" Two reasons come to mind:

1. Windows does not (yet) enjoy a large community of developers who are willing to freely share their source. As a direct result, the necessary investment in infrastructure required to support such development hasn't been made. By and large, what is available has been made possible by the porting of necessary utilities from Unix. Don't be surprised if some of this heritage shows through from time to time.
2. Pretty much all of the instructions that follow are of the "set and forget" variety. So sit back and try follow the instructions below as faithfully as you can.

Preparations

Before you get started, you have a lot to download...

- For starters, get the Cygwin toolkit from the closest cygwin (http://sources.redhat.com/cygwin/download.html) mirror site. This will provide you most of the popular GNU utilities used by the build process.

- Download the rest of the build tools you will need from the PHP site at http://www.php.net/extra/win32build.zip.
- Get the source code for the DNS name resolver used by PHP at http://www.php.net/extra/bindlib_w32.zip. This is a replacement for the `resolv.lib` library included in `win32build.zip`.
- If you don't already have an unzip utility, you will need one. A free version is available from InfoZip (http://www.cdrom.com/pub/infozip/UnZip.html).

Finally, you are going to need the source to PHP 4 itself. You can get the latest development version using anonymous CVS (http://www.php.net/anoncvs.php). If you get a snapshot (http://snaps.php.net/) or a source (http://www.php.net/downloads.php) tarball, you not only will have to untar and ungzip it, but you will have to convert the bare linefeeds to crlf's in the `*.dsp` and `*.dsw` files before Microsoft Visual C++ will have anything to do with them.

> **Nota:** Place the `Zend` and `TSRM` directories inside the `php4` directory in order for the projects to be found during the build process.

Putting it all together

- Follow the instructions for installing the unzip utility of your choosing.
- Execute `setup.exe` and follow the installation instructions. If you choose to install to a path other than `c:\cygnus`, let the build process know by setting the Cygwin environment variable. On Windows 95/98 setting an environment variable can be done by placing a line in your `autoexec.bat`. On Windows NT, go to My Computer => Control Panel => System and select the environment tab.

> **Attenzione**
>
> Make a temporary directory for Cygwin to use, otherwise many commands (particularly bison) will fail. On Windows 95/98, **`mkdir C:\TMP`**. For Windows NT, **`mkdir %SystemDrive%\tmp`**.

- Make a directory and unzip `win32build.zip` into it.
- Launch Microsoft Visual C++, and from the menu select Tools => Options. In the dialog, select the directories tab. Sequentially change the dropdown to Executables, Includes, and Library files, and ensure that `cygwin\bin`, `win32build\include`, and `win32build\lib` are in each list, respectively. (To add an entry, select a blank line at the end of the list and begin typing). Typical entries will look like this:
  - `c:\cygnus\bin`
  - `c:\php-win32build\include`
  - `c:\php-win32build\lib`

  Press OK, and exit out of Visual C++.

- Make another directory and unzip `bindlib_w32.zip` into it. Decide whether you want to have debug symbols available (bindlib - Win32 Debug) or not (bindlib - Win32 Release). Build the appropriate configuration:
  - For GUI users, launch VC++, and then select File => Open Workspace and select bindlib. Then select Build=>Set Active Configuration and select the desired configuration. Finally select Build=>Rebuild All.
  - For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following:
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Debug"`**
    - **`msdev bindlib.dsp /MAKE "bindlib - Win32 Release"`**
  - At this point, you should have a usable `resolv.lib` in either your `Debug` or `Release` subdirectories. Copy this file into your `win32build\lib` directory over the file by the same name found in there.

Compiling

The best way to get started is to build the standalone/CGI version.

- For GUI users, launch VC++, and then select File => Open Workspace and select php4ts. Then select Build=>Set Active Configuration and select the desired configuration. Finally select Build=>Rebuild All.
- For command line users, make sure that you either have the C++ environment variables registered, or have run **vcvars.bat**, and then execute one of the following:
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Debug_TS"`**
  - **`msdev php4ts.dsp /MAKE "php4ts - Win32 Release_TS"`**
  - At this point, you should have a usable `php.exe` in either your `Debug_TS` or `Release_TS` subdirectories.

Repeat the above steps with `php4isapi.dsp` (which can be found in `sapi\isapi`) in order to build the code necessary for integrating PHP with Microsoft IIS.

Installation of Windows extensions

After installing PHP and a webserver on Windows, you will probably want to install some extensions for added functionality. The following table describes some of the extensions available. You can choose which extensions you would like to load when PHP starts by uncommenting the: ’extension=php_*.dll’ lines in `php.ini`. You can also load a module dynamically in your script using dl().

The DLLs for PHP extensions are prefixed with ’php_’ in PHP 4 (and ’php3_’ in PHP 3). This prevents confusion between PHP extensions and their supporting libraries.

Nota: In PHP 4.0.6 BCMath, Calendar, COM, FTP, MySQL, ODBC, PCRE, Session, WDDX and XML support is *built in*. You don't need to load any additional extensions in order to use these functions. See your distributions `README.txt` or `install.txt` for a list of built in modules.

Nota: Some of these extensions need extra DLLs to work. Couple of them can be found in the distribution package, in the 'dlls' folder but some, for example Oracle (php_oci8.dll) require DLLs which are not bundled with the distribution package.

Copy the bundled DLLs from 'DLLs' folder to your Windows PATH, safe places are:

c:\windows\system for Windows 9x/Me
c:\winnt\system32 for Windows NT/2000
c:\windows\system32 for Windows XP

If you have them already installed on your system, overwrite them only if something doesn't work correctly (Before overwriting them, it is a good idea to make a backup of them, or move them to another folder - just in case something goes wrong).

Tabella 2-1. PHP Extensions

Extension	Description	Notes
php_bz2.dll	bzip2 compression functions	None
php_calendar.dll	Calendar conversion functions	Built in since PHP 4.0.3
php_cpdf.dll	ClibPDF functions	None
php_crack.dll	Crack functions	None
php3_crypt.dll	Crypt functions	unknown
php_ctype.dll	ctype family functions	None
php_curl.dll	CURL, Client URL library functions	Requires: libeay32.dll, ssleay32.dll (bundled)
php_cybercash.dll	Cybercash payment functions	None
php_db.dll	DBM functions	Deprecated. Use DBA instead (php_dba.dll)
php_dba.dll	DBA: DataBase (dbm-style) Abstraction layer functions	None
php_dbase.dll	dBase functions	None
php3_dbm.dll	Berkeley DB2 library	unknown
php_domxml.dll	DOM XML functions	Requires: libxml2.dll (bundled)
php_dotnet.dll	.NET functions	None
php_exif.dll	Read EXIF headers from JPEG	None
php_fbsql.dll	FrontBase functions	None
php_fdf.dll	FDF: Forms Data Format functions.	Requires: fdftk.dll (bundled)
php_filepro.dll	filePro functions	Read-only access
php_ftp.dll	FTP functions	Built-in since PHP 4.0.3
php_gd.dll	GD library image functions	None

Extension	Description	Notes
php_gettext.dll	Gettext functions	Requires: gnu_gettext.dll (bundled)
php_hyperwave.dll	HyperWave functions	None
php_iconv.dll	ICONV characterset conversion	Requires: iconv-1.3.dll (bundled)
php_ifx.dll	Informix functions	Requires: Informix libraries
php_iisfunc.dll	IIS management functions	None
php_imap.dll	IMAP POP3 and NNTP functions	PHP 3: php3_imap4r1.dll
php_ingres.dll	Ingres II functions	Requires: Ingres II libraries
php_interbase.dll	InterBase functions	Requires: gds32.dll (bundled)
php_java.dll	Java extension	Requires: jvm.dll (bundled)
php_ldap.dll	LDAP functions	Requires: libsasl.dll (bundled)
php_mhash.dll	Mhash Functions	None
php_ming.dll	Ming functions for Flash	None
php_msql.dll	mSQL functions	Requires: msql.dll (bundled)
php3_msql1.dll	mSQL 1 client	unknown
php3_msql2.dll	mSQL 2 client	unknown
php_mssql.dll	MSSQL functions	Requires: ntwdblib.dll (bundled)
php3_mysql.dll	MySQL functions	Built-in in PHP 4
php3_nsmail.dll	Netscape mail functions	unknown
php3_oci73.dll	Oracle functions	unknown
php_oci8.dll	Oracle 8 functions	Requires: Oracle 8 client libraries
php_openssl.dll	OpenSSL functions	Requires: libeay32.dll (bundled)
php_oracle.dll	Oracle functions	Requires: Oracle 7 client libraries
php_pdf.dll	PDF functions	None
php_pgsql.dll	PostgreSQL functions	None
php_printer.dll	Printer functions	None
php_xslt.dll	XSLT functions	Requires: sablot.dll (bundled)
php_snmp.dll	SNMP get and walk functions	NT only!
php_sybase_ct.dll	Sybase functions	Requires: Sybase client libraries
php_yaz.dll	YAZ functions	None
php_zlib.dll	ZLib compression functions	None

Servers-CGI/Commandline

The default is to build PHP as a CGI program. This creates a commandline interpreter, which can be used for CGI processing, or for non-web-related PHP scripting. If you are running a web server PHP

has module support for, you should generally go for that solution for performance reasons. However, the CGI version enables Apache users to run different PHP-enabled pages under different user-ids. Please make sure you read through the Security chapter if you are going to run PHP as a CGI.

Testing

If you have built PHP as a CGI program, you may test your build by typing **make test**. It is always a good idea to test your build. This way you may catch a problem with PHP on your platform early instead of having to struggle with it later.

Benchmarking

If you have built PHP 3 as a CGI program, you may benchmark your build by typing **make bench**. Note that if Safe Mode is on by default, the benchmark may not be able to finish if it takes longer then the 30 seconds allowed. This is because the set_time_limit() can not be used in safe mode. Use the max_execution_time configuration setting to control this time for your own scripts. **make bench** ignores the configuration file.

> **Nota: make bench** is only available for PHP 3.

Servers-Apache

This section contains notes and hints specific to Apache installs of PHP, both for Unix and Windows versions.

Details of installing PHP with Apache on Unix

You can select arguments to add to the **configure** on line 8 below from the Complete list of configure options. The version numbers have been omitted here, to ensure the instructions are not incorrect. You will need to replace the ’xxx’ here with the correct values from your files.

Esempio 2-5. Installation Instructions (Apache Shared Module Version) for PHP 4

```
1.  gunzip apache_xxx.tar.gz
2.  tar -xvf apache_xxx.tar
3.  gunzip php-xxx.tar.gz
4.  tar -xvf php-xxx.tar
5.  cd apache_xxx
6.  ./configure --prefix=/www --enable-module=so
7.  make
8.  make install
9.  cd ../php-xxx
10. ./configure --with-mysql --with-apxs=/www/bin/apxs
11. make
12. make install
```

```
  If you decide to change your configure options after installation
  you only need to repeat the last three steps. You only need to
  restart apache for the new module to take effect. A recompile of
  Apache is not needed.

11. cp php.ini-dist /usr/local/lib/php.ini

  You can edit your .ini file to set PHP options.  If
  you prefer this file in another location, use
  --with-config-file-path=/path in step 8.

12. Edit your httpd.conf or srm.conf file and check that these lines are
    present and not commented out:

   AddType application/x-httpd-php .php

   LoadModule php4_module        libexec/libphp4.so

  You can choose any extension you wish here.  .php is simply the one
  we suggest. You can even include .html, and .php3 can be added for
  backwards compatibility.

  The path on the right hand side of the LoadModule statement must point
  to the path of the PHP module on your system. The above statement is
  correct for the steps shown above.


13. Use your normal procedure for starting the Apache server. (You must
    stop and restart the server, not just cause the server to reload by
    use a HUP or USR1 signal.)
```

Depending on your Apache install and Unix variant, there are many possible ways to stop and restart the server. Below are some typical lines used in restarting the server, for different apache/unix installations. You should replace `/path/to/` with the path to these applications on your systems.

```
1. Several Linux and SysV variants:
/etc/rc.d/init.d/httpd restart

2. Using apachectl scripts:
/path/to/apachectl stop
/path/to/apachectl start

3. httpdctl and httpsdctl (Using OpenSSL), similar to apachectl:
/path/to/httpsdctl stop
/path/to/httpsdctl start

4. Using mod_ssl, or another SSL server, you may want to manually
stop and start:
/path/to/apachectl stop
/path/to/apachectl startssl
```

The locations of the apachectl and http(s)dctl binaries often vary. If your system has `locate` or `whereis` or `which` commands, these can assist you in finding your server control programs.

Different examples of compiling PHP for apache are as follows:

```
./configure --with-apxs --with-pgsql
```

This will create a `libphp4.so` shared library that is loaded into Apache using a LoadModule line in Apache's `httpd.conf` file. The PostgreSQL support is embedded into this `libphp4.so` library.

```
./configure --with-apxs --with-pgsql=shared
```

This will create a `libphp4.so` shared library for Apache, but it will also create a `pgsql.so` shared library that is loaded into PHP either by using the extension directive in `php.ini` file or by loading it explicitly in a script using the dl() function.

```
./configure --with-apache=/path/to/apache_source --with-pgsql
```

This will create a `libmodphp4.a` library, a `mod_php4.c` and some accompanying files and copy this into the `src/modules/php4` directory in the Apache source tree. Then you compile Apache using `--activate-module=src/modules/php4/libphp4.a` and the Apache build system will create `libphp4.a` and link it statically into the `httpd` binary. The PostgreSQL support is included directly into this `httpd` binary, so the final result here is a single `httpd` binary that includes all of Apache and all of PHP.

```
./configure --with-apache=/path/to/apache_source --with-pgsql=shared
```

Same as before, except instead of including PostgreSQL support directly into the final `httpd` you will get a `pgsql.so` shared library that you can load into PHP from either the `php.ini` file or directly using dl().

When choosing to build PHP in different ways, you should consider the advantages and drawbacks of each method. Building as a shared object will mean that you can compile apache separately, and don't have to recompile everything as you add to, or change, PHP. Building PHP into apache (static method) means that PHP will load and run faster. For more information, see the Apache webpage on DSO support (http://www.apache.org/docs/dso.html).

Nota: Apache's default http.conf currently ships with a section that looks like this: User nobody Group "#-1" Unless you change that to "Group nogroup" or something like that ("Group daemon" is also very common) PHP will not be able to open files.

Installing PHP on Windows with Apache 1.3.x

There are two ways to set up PHP to work with Apache 1.3.x on Windows. One is to use the CGI binary (php.exe), the other is to use the Apache module DLL. In either case you need to stop the Apache server, and edit your `srm.conf` or `httpd.conf` to configure Apache to work with PHP.

It is worth noting here that now the SAPI module has been made more stable under windows, we recommend it's use above the CGI binary, since it is more transparent and secure.

Although there can be a few variations of configuring PHP under Apache, these are simple enough to be used by the newcomer. Please consult the Apache Docs for further configuration directives.

If you unziped the PHP package to c:\php\ as described in the Manual Installation Steps section, you need to insert these lines to your Apache configuration file to set up the CGI binary:

- `ScriptAlias /php/ "c:/php/"`
- `AddType application/x-httpd-php .php .phtml`
- `Action application/x-httpd-php "/php/php.exe"`

Note that the second line in the list above can be found in the actual versions of `httpd.conf`, but it is commented out. Remember also to substitute the `c:/php/` for your actual path to PHP.

Attenzione

By using the CGI setup, your server is open to several possible attacks. Please read our CGI security section to learn how to defend yourself from attacks.

If you would like to use PHP as a module in Apache, be sure to move `php4ts.dll` to the windows/system (for Windows 9x/Me) or winnt/system32 (for Windows NT/2000/XP) directory, overwriting any older file. Then you should add the following two lines to you Apache conf file:

- `LoadModule php4_module c:/php/sapi/php4apache.dll`
- `AddType application/x-httpd-php .php .phtml`

After changing the configuration file, remember to restart the server, for example, `NET STOP APACHE` followed by `NET START APACHE`, if you run Apache as a Windows Service, or use your regular shortcuts.

Nota: You may find after using the windows installer for Apache that you need to define the `AddModule` directive for `mod_php4.c` in the configuration file (`httpd.conf`). This is done by adding `AddModule mod_php4.c` to the `AddModule` list, near the beginning of the configuration file. This is especially important if the `ClearModuleList` directive is defined. Failure to do this may mean PHP will not be registered as an Apache module.

There are 2 ways you can use the source code highlighting feature, however their ability to work depends on your installation. If you have configured Apache to use PHP as an ISAPI module, then by adding the following line to your configuration file you can use this feature: `AddType application/x-httpd-php-source .phps`

If you chose to configure Apache to use PHP as a CGI binary, you will need to use the show_source() function. To do this simply create a PHP script file and add this code: `<?php show_source ("original_php_script.php"); ?>`. Substitute `original_php_script.php` with the name of the file you wish to show the source of.

> **Nota:** On Win-Apache all backslashes in a path statement such as "c:\directory\file.ext", must be converted to forward slashes, as "c:/directory/file.ext".

Servers-Caudium

PHP 4 can be built as a Pike module for the Caudium webserver. Note that this is not supported with PHP 3. Follow the simple instructions below to install PHP 4 for Caudium.

Esempio 2-6. Caudium Installation Instructions

```
1.  Make sure you have Caudium installed prior to attempting to
    install PHP 4. For PHP 4 to work correctly, you will need Pike
    7.0.268 or newer. For the sake of this example we assume that
    Caudium is installed in /opt/caudium/server/.
2.  Change directory to php-x.y.z (where x.y.z is the version number).
3.  ./configure --with-caudium=/opt/caudium/server
4.  make
5.  make install
6.  Restart Caudium if it’s currently running.
7.  Log into the graphical configuration interface and go to the
    virtual server where you want to add PHP 4 support.
8.  Click Add Module and locate and then add the PHP 4 Script Support module.
9.  If the documentation says that the ’PHP 4 interpreter isn’t
    available’, make sure that you restarted the server. If you did
    check /opt/caudium/logs/debug/default.1 for any errors related to
    <filename>PHP4.so</filename>. Also make sure that
    <filename>caudium/server/lib/[pike-version]/PHP4.so</filename>
    is present.
10. Configure the PHP Script Support module if needed.
```

You can of course compile your Caudium module with support for the various extensions available in PHP 4. See the complete list of configure options for an exhaustive rundown.

> **Nota:** When compiling PHP 4 with MySQL support you must make sure that the normal MySQL client code is used. Otherwise there might be conflicts if your Pike already has MySQL support. You do this by specifying a MySQL install directory the --with-mysql option.

Servers-fhttpd

To build PHP as an fhttpd module, answer "yes" to "Build as an fhttpd module?" (the --with-fhttpd=*DIR* option to configure) and specify the fhttpd source base directory. The default directory is `/usr/local/src/fhttpd`. If you are running fhttpd, building PHP as a module will give better performance, more control and remote execution capability.

Servers-IIS/PWS

This section contains notes and hints specific to IIS (Microsoft Internet Information Server). Installing PHP for PWS/IIS 3, PWS 4 or newer and IIS 4 or newer versions.

Windows and PWS/IIS 3

The recommended method for configuring these servers is to use the REG file included with the distribution (pws-php4cgi.reg). You may want to edit this file and make sure the extensions and PHP install directories match your configuration. Or you can follow the steps below to do it manually.

> **Attenzione**
>
> These steps involve working directly with the Windows registry. One error here can leave your system in an unstable state. We highly recommend that you back up your registry first. The PHP Development team will not be held responsible if you damage your registry.

- Run Regedit.
- Navigate to: `HKEY_LOCAL_MACHINE /System /CurrentControlSet /Services /W3Svc /Parameters /ScriptMap`.
- On the edit menu select: `New->String Value`.
- Type in the extension you wish to use for your php scripts. For example `.php`
- Double click on the new string value and enter the path to `php.exe` in the value data field. ex: `c:\php\php.exe`.
- Repeat these steps for each extension you wish to associate with PHP scripts.

The following steps do not affect the web server installation and only apply if you want your php scripts to be executed when they are run from the command line (ex. run `c:\myscripts\test.php`) or by double clicking on them in a directory viewer window. You may wish to skip these steps as you might prefer the PHP files to load into a text editor when you double click on them.

- Navigate to: `HKEY_CLASSES_ROOT`
- On the edit menu select: `New->Key`.
- Name the key to the extension you setup in the previous section. ex: `.php`

- Highlight the new key and in the right side pane, double click the "default value" and enter `phpfile`.
- Repeat the last step for each extension you set up in the previous section.
- Now create another `New->Key` under `HKEY_CLASSES_ROOT` and name it `phpfile`.
- Highlight the new key `phpfile` and in the right side pane, double click the "default value" and enter `PHP Script`.
- Right click on the `phpfile` key and select `New->Key`, name it `Shell`.
- Right click on the `Shell` key and select `New->Key`, name it `open`.
- Right click on the `open` key and select `New->Key`, name it `command`.
- Highlight the new key `command` and in the right side pane, double click the "default value" and enter the path to `php.exe`. ex: `c:\php\php.exe -q %1`. (don't forget the `%1`).
- Exit Regedit.
- If using PWS on Windows, reboot to reload the registry.

PWS and IIS 3 users now have a fully operational system. IIS 3 users can use a nifty tool (http://www.genusa.com/iis/iiscfg.html) from Steven Genusa to configure their script maps.

Windows and PWS 4 or newer

When installing PHP on Windows with PWS 4 or newer version, you have two options. One to set up the PHP CGI binary, the other is to use the ISAPI module DLL.

If you choose the CGI binary, do the following:

- Edit the enclosed `pws-php4cgi.reg` file (look into the SAPI dir) to reflect the location of your php.exe. Forward slashes should be escaped, for example: `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="c:\\php\\php.exe"`
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

If you choose the ISAPI module, do the following:

- Edit the enclosed `pws-php4isapi.reg` file (look into the SAPI dir) to reflect the location of your php4isapi.dll. Forward slashes should be escaped, for example: `[HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\w3svc\parameters\Script Map] ".php"="c:\\php\\sapi\\php4isapi.dll"`
- In the PWS Manager, right click on a given directory you want to add PHP support to, and select Properties. Check the 'Execute' checkbox, and confirm.

Windows NT/2000/XP and IIS 4 or newer

To install PHP on an NT/2000/XP Server running IIS 4 or newer, follow these instructions. You have two options to set up PHP, using the CGI binary (php.exe) or with the ISAPI module.

In either case, you need to start the Microsoft Management Console (may appear as 'Internet Services Manager', either in your Windows NT 4.0 Option Pack branch or the Control Panel=>Administrative Tools under Windows 2000/XP). Then right click on your Web server node (this will most probably appear as 'Default Web Server'), and select 'Properties'.

If you want to use the CGI binary, do the following:

- Under 'Home Directory', 'Virtual Directory', or 'Directory', click on the 'Configuration' button, and then enter the App Mappings tab.
- Click Add, and in the Executable box, type: `c:\php\php.exe` (assuming that you have unziped PHP in c:\php\).
- In the Extension box, type the file name extension you want associated with PHP scripts. Leave 'Method exclusions' blank, and check the Script engine checkbox. You may also like to check the 'check that file exists' box - for a small performance penalty, IIS (or PWS) will check that the script file exists and sort out authentication before firing up php. This means that you will get sensible 404 style error messages instead of cgi errors complaining that php did not output any data.

  You must start over from the previous step for each extension you want associated with PHP scripts. `.php` and `.phtml` are common, although `.php3` may be required for legacy applications.
- Set up the appropriate security. (This is done in Internet Service Manager), and if your NT Server uses NTFS file system, add execute rights for I_USR_ to the directory that contains `php.exe`.

To use the ISAPI module, do the following:

- If you don't want to perform HTTP Authentication using PHP, you can (and should) skip this step. Under ISAPI Filters, add a new ISAPI filter. Use PHP as the filter name, and supply a path to the php4isapi.dll.
- Under 'Home Directory', click on the 'Configuration' button. Add a new entry to the Application Mappings. Use the path to the php4isapi.dll as the Executable, supply `.php` as the extension, leave Method exclusions blank, and check the Script engine checkbox.
- Stop IIS completely (NET STOP iisadmin)
- Start IIS again (NET START w3svc)

Servers-Netscape and iPlanet

This section contains notes and hints specific to Netscape and iPlanet installs of PHP, both for Sun Solaris and Windows versions.

You can find more information about setting up PHP for the Netscape Enterprise Server here: http://benoit.noss.free.fr/php/install-php4.html

Installing PHP with Netscape on Sun Solaris

To build PHP with NES or iPlanet web servers, enter the proper install directory for the --with-nsapi = *DIR* option. The default directory is usually `/opt/netscape/suitespot/`. Please also read `/php-xxx-version/sapi/nsapi/nsapi-readme.txt`.

Esempio 2-7. Installation Example for Netscape Enterprise on Solaris

```
Instructions for Sun Solaris 2.6 with Netscape Enterprise Server 3.6
From: bhager@invacare.com

1. Install the following packages from www.sunfreeware.com or another
download site:

    flex-2_5_4a-sol26-sparc-local
    gcc-2_95_2-sol26-sparc-local
    gzip-1.2.4-sol26-sparc-local
    perl-5_005_03-sol26-sparc-local
    bison-1_25-sol26-sparc-local
    make-3_76_1-sol26-sparc-local
    m4-1_4-sol26-sparc-local
    autoconf-2.13
    automake-1.4
    mysql-3.23.24-beta (if you want mysql support)
    tar-1.13 (GNU tar)

2. Make sure your path includes the proper directories
    PATH=.:/usr/local/bin:/usr/sbin:/usr/bin:/usr/ccs/bin
    export PATH

3. gunzip php-x.x.x.tar.gz (if you have a .gz dist, otherwise go to 4)
4. tar xvf php-x.x.x.tar
5. cd ../php-x.x.x

6. For the following step, make sure /opt/netscape/suitespot/ is where
   your netscape server is installed. Otherwise, change to correct path:
    /configure --with-mysql=/usr/local/mysql --with-nsapi=/opt/netscape/suitespot/ -
-enable-track-vars --enable-libgcc
7. make
8. make install
```

After performing the base install and reading the appropriate readme file, you may need to performs some additional configuration steps.

Firstly you may need to add some paths to the LD_LIBRARY_PATH environment for Netscape to find all the shared libs. This can best done in the start script for your Netscape server. Windows users can probably skip this step. The start script is often located in: `/path/to/server/https-servername/start`

You may also need to edit the configuration files that are located in:`/path/to/server/https-servername/config/`.

Esempio 2-8. Configuration Example for Netscape Enterprise

```
Configuration Instructions for Netscape Enterprise Server
From: bhager@invacare.com

1. Add the following line to mime.types:
    type=magnus-internal/x-httpd-php exts=php

2. Add the following to obj.conf, shlib will vary depending on
    your OS, for unix it will be something like
    /opt/netscape/suitespot/bin/libphp4.so.

    You should place the following lines after mime types init.
    Init fn="load-modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shli
    Init fn=php4_init errorString="Failed to initialize PHP!"

    <object name="default">
    .
    .
    .
    .#NOTE this next line should happen after all 'ObjectType' and before all 'AddLog' li
    Service fn="php4_execute" type="magnus-internal/x-httpd-php"
    .
    .
    </Object>


    <Object name="x-httpd-php">
    ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
    Service fn=php4_execute
    </Object>


    Authentication configuration

    PHP authentication cannot be used with any other authentication. ALL AU-
THENTICATION IS
    PASSED TO YOUR PHP SCRIPT. To configure PHP Authentication for the entire server, add
    the following line:

    <Object name="default">
    AuthTrans fn=php4_auth_trans
    .
    .
    .
    .
    </Object>

    To use PHP Authentication on a single directory, add the following:

    <Object ppath="d:\path\to\authenticated\dir\*">
    AuthTrans fn=php4_auth_trans
    </Object>
```

If you are running Netscape Enterprise 4.x, then you should use the following:

Esempio 2-9. Configuration Example for Netscape Enterprise 4.x

```
Place these lines after the mime types init, and everything else is similar
to the example configuration above.
From: Graeme Hoose (GraemeHoose@BrightStation.com)

Init fn="load-modules" shlib="/path/to/server4/bin/libphp4.so" funcs="php4_init,php4_clos
Init fn="php4_init" LateInit="yes"
```

Installing PHP with Netscape on Windows

To Install PHP as CGI (for Netscape Enterprise Server, iPlanet, perhaps Fastrack), do the following:

- Copy `php4ts.dll` to your systemroot (the directory where you installed windows)
- Make a file association from the command line. Type the following two lines:

  ```
  assoc .php=PHPScript
  ftype PHPScript=c:\php\php.exe %1 %*
  ```

- In the Netscape Enterprise Administration Server create a dummy shellcgi directory and remove it just after (this step creates 5 important lines in obj.conf and allow the web server to handle shellcgi scripts).
- In the Netscape Enterprise Administration Server create a new mime type (Category: type, Content-Type: magnus-internal/shellcgi, File Suffix:php).
- Do it for each web server instance you want php to run

More details about setting up PHP as a CGI executable can be found here: http://benoit.noss.free.fr/php/install-php.html

To Install PHP as NSAPI (for Netscape Enterprise Server, iPlanet, perhaps Fastrack, do the following:

- Copy `php4ts.dll` to your systemroot (the directory where you installed windows)
- Make a file association from the command line. Type the following two lines:

  ```
  assoc .php=PHPScript
  ftype PHPScript=c:\php\php.exe %1 %*
  ```

- In the Netscape Enterprise Administration Server create a new mime type (Category: type, Content-Type: magnus-internal/x-httpd-php, File Suffix:php).
- Stop your web service and edit `obj.conf`. At the end of the Init section, place these two lines (necessarily after mime type init!):

```
Init fn="load-modules" funcs="php4_init,php4_close,php4_execute,php4_auth_trans" shlib=
Init fn="php4_init" errorString="Failed to initialise PHP!"
```

- In The `< Object name="default" >` section, place this line necessarily after all 'ObjectType' and before all 'AddLog' lines:

```
Service fn="php4_execute" type="magnus-internal/x-httpd-php"
```

- At the end of the file, create a new object called `x-httpd-php`, by inserting these lines:

```
<Object name="x-httpd-php">
ObjectType fn="force-type" type="magnus-internal/x-httpd-php"
Service fn=php4_execute
</Object>
```

- Restart your web service and apply changes
- Do it for each web server instance you want PHP to run

More details about setting up PHP as an NSAPI filter can be found here:
http://benoit.noss.free.fr/php/install-php4.html

Servers-OmniHTTPd Server

This section contains notes and hints specific to OmniHTTPd.

OmniHTTPd 2.0b1 and up for Windows

You need to complete the following steps to make PHP work with OmniHTTPd. This is a CGI executable setup. SAPI is supported by OmniHTTPd, but some tests have shown that it is not so stable to use PHP as an ISAPI module.

- Step 1: Install OmniHTTPd server.
- Step 2: Right click on the blue OmniHTTPd icon in the system tray and select `Properties`
- Step 3: Click on `Web Server Global Settings`
- Step 4: On the 'External' tab, enter: `virtual = .php | actual = c:\path-to-php-dir\php.exe`, and use the Add button.
- Step 5: On the `Mime` tab, enter: `virtual = wwwserver/stdcgi | actual = .php`, and use the Add button.
- Step 6: Click `OK`

Repeat steps 2 - 6 for each extension you want to associate with PHP.

> **Nota:** Some OmniHTTPd packages come with built in PHP support. You can choose at setup time to do a custom setup, and uncheck the PHP component. We recommend you to use the latest PHP binaries. Some OmniHTTPd servers come with PHP 4 beta distributions, so you should choose not to set up the built in support, but install your own. If the server is already on your machine, use the Replace button in Step 4 and 5 to set the new, correct information.

Servers-Oreilly Website Pro

This section contains notes and hints specific to Oreilly Website Pro.

Oreilly Website Pro 2.5 and up for Windows

This list describes how to set up the PHP CGI binary or the ISAPI module to work with Oreilly Website Pro on Windows.

- Edit the Server Properties and select the tab "Mapping".
- From the List select "Associations" and enter the desired extension (`.php`) and the path to the CGI exe (ex. `c:\php\php.exe`) or the ISAPI DLL file (ex. `c:\php\sapi\php4isapi.dll`).
- Select "Content Types" add the same extension (`.php`) and enter the content type. If you choose the CGI executable file, enter ’wwwserver/shellcgi’, if you choose the ISAPI module, enter ’wwwserver/isapi’ (both without quotes).

Servers-Xitami

This section contains notes and hints specific to Xitami.

Xitami for Windows

This list describes how to set up the PHP CGI binary to work with Xitami on Windows.

- Make sure the webserver is running, and point your browser to xitamis admin console (usually `http://127.0.0.1/admin`), and click on Configuration.
- Navigate to the Filters, and put the extension which PHP should parse (i.e. .php) into the field File extensions (.xxx).
- In Filter command or script put the path and name of your php executable i.e. `c:\php\php.exe`.
- Press the ’Save’ icon.
- Restart the server to reflect changes.

Servers-Other web servers

PHP can be built to support a large number of web servers. Please see Server-related options for a full list of server-related configure options. The PHP CGI binaries are compatible with almost all webservers supporting the CGI standard.

Problems?

Read the FAQ

Some problems are more common than others. The most common ones are listed in the PHP FAQ, part of this manual.

Other problems

If you are still stuck, someone on the PHP installation mailing list may be able to help you. You should check out the archive first, in case someone already answered someone else who had the same problem as you. The archives are available from the support page on http://www.php.net/. To subscribe to the PHP installation mailing list, send an empty mail to php-install-subscribe@lists.php.net (mailto:php-install-subscribe@lists.php.net). The mailing list address is `php-install@lists.php.net`.

If you want to get help on the mailing list, please try to be precise and give the necessary details about your environment (which operating system, what PHP version, what web server, if you are running PHP as CGI or a server module, etc.), and preferably enough code to make others able to reproduce and test your problem.

Bug reports

If you think you have found a bug in PHP, please report it. The PHP developers probably don't know about it, and unless you report it, chances are it won't be fixed. You can report bugs using the bug-tracking system at http://bugs.php.net/. Please do not send bug reports in mailing list or personal letters. The bug system is also suitable to submit feature requests.

Read the How to report a bug (http://bugs.php.net/how-to-report.php) document before submitting any bug reports!

Complete list of configure options

> **Nota:** These are only used at compile time. If you want to alter PHP's runtime configuration, please see the chapter on Configuration.

The following is a complete list of options supported by PHP 4 `configure` scripts (as of 4.1.0), used when compiling in Unix-like environments. Some are available in PHP 3, some in PHP 4, and some in both. This is not noted yet.

- Database
- Graphics
- Miscellaneous
- PHP Behaviour
- Server
- XML

Configure Options in PHP 4

> **Nota:** These options are only used in PHP 4 as of PHP 4.1.0. Some are available in older versions of PHP 4, some even in PHP 3, some only in PHP 4.1.0. If you want to compile an older version, some options will probably not be available.

Database options

--with-db

Include old xDBM support (deprecated).

--enable-dba=shared

Build DBA as a shared module.

--with-gdbm[=DIR]

Include GDBM support.

--with-ndbm[=DIR]

Include NDBM support.

--with-db2[=DIR]

Include Berkeley DB2 support.

--with-db3[=DIR]

Include Berkeley DB3 support.

--with-dbm[=DIR]

Include DBM support.

--with-cdb[=DIR]

Include CDB support.

--enable-dbase

Enable the bundled dbase library.

--with-dbplus

Include dbplus support.

--enable-dbx

Enable dbx.

--with-fbsql[=DIR]

Include FrontBase support. DIR is the FrontBase base directory.

--enable-filepro

Enable the bundled read-only filePro support.

--with-fribidi[=DIR]

Include fribidi support (requires FriBidi >=0.1.12). DIR is the fribidi installation directory - default /usr/local/.

--with-informix[=DIR]

Include Informix support. DIR is the Informix base install directory, defaults to nothing.

--with-ingres[=DIR]

Include Ingres II support. DIR is the Ingres base directory (default /II/ingres).

--with-interbase[=DIR]

Include InterBase support. DIR is the InterBase base install directory, defaults to /usr/interbase.

--with-msql[=DIR]

Include mSQL support. DIR is the mSQL base install directory, defaults to /usr/local/Hughes.

--with-mysql[=DIR]

Include MySQL support. DIR is the MySQL base directory. If unspecified, the bundled MySQL library will be used.

--with-oci8[=DIR]

Include Oracle-oci8 support. Default DIR is ORACLE_HOME.

--with-adabas[=DIR]

Include Adabas D support. DIR is the Adabas base install directory, defaults to /usr/local.

--with-sapdb[=DIR]

Include SAP DB support. DIR is SAP DB base install directory, defaults to /usr/local.

--with-solid[=DIR]

Include Solid support. DIR is the Solid base install directory, defaults to /usr/local/solid.

--with-ibm-db2[=DIR]

Include IBM DB2 support. DIR is the DB2 base install directory, defaults to /home/db2inst1/sqllib.

--with-empress[=DIR]

Include Empress support. DIR is the Empress base install directory, defaults to $EMPRESSPATH. From PHP4, this option only supports Empress Version 8.60 and above.

--with-empress-bcs[=DIR]

Include Empress Local Access support. DIR is the Empress base install directory, defaults to $EMPRESSPATH. From PHP4, this option only supports Empress Version 8.60 and above.

--with-birdstep[=DIR]

Include Birdstep support. DIR is the Birdstep base install directory, defaults to /usr/local/birdstep.

--with-custom-odbc[=DIR]

Include a user defined ODBC support. The DIR is ODBC install base directory, which defaults to /usr/local. Make sure to define CUSTOM_ODBC_LIBS and have some odbc.h in your include dirs. E.g., you should define following for Sybase SQL Anywhere 5.5.00 on QNX, prior to run configure script: CPPFLAGS="-DODBC_QNX -DSQLANY_BUG" LDFLAGS=-lunix CUSTOM_ODBC_LIBS="-ldblib -lodbc".

--with-iodbc[=DIR]

Include iODBC support. DIR is the iODBC base install directory, defaults to /usr/local.

--with-esoob[=DIR]

Include Easysoft OOB support. DIR is the OOB base install directory, defaults to /usr/local/easysoft/oob/client.

--with-unixODBC[=DIR]

Include unixODBC support. DIR is the unixODBC base install directory, defaults to /usr/local.

--with-openlink[=DIR]

Include OpenLink ODBC support. DIR is the OpenLink base install directory, defaults to /usr/local. This is the same as iODBC.

--with-dbmaker[=DIR]

Include DBMaker support. DIR is the DBMaker base install directory, defaults to where the latest version of DBMaker is installed (such as /home/dbmaker/3.6).

--with-oracle[=DIR]

Include Oracle-oci7 support. Default DIR is ORACLE_HOME.

--with-ovrimos[=DIR]

Include Ovrimos SQL Server support. DIR is the Ovrimos’ libsqlcli install directory.

--with-pgsql[=DIR]

Include PostgreSQL support. DIR is the PostgreSQL base install directory, defaults to /usr/local/pgsql. Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

--with-sybase[=DIR]

Include Sybase-DB support. DIR is the Sybase home directory, defaults to /home/sybase.

--with-sybase-ct[=DIR]

Include Sybase-CT support. DIR is the Sybase home directory. Defaults to /home/sybase.

--disable-unified-odbc

Disable unified ODBC support. Only applicable if iODBC, Adabas, Solid, Velocis or a custom ODBC interface is enabled. PHP 3 only!

Graphics options

--with-gd[=DIR]

Include GD support (DIR is GD's install dir). Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

--enable-gd-native-ttf

GD: Enable TrueType string function in gd.

--with-jpeg-dir=DIR

GD: Set the path to libjpeg install prefix.

--with-png-dir=DIR

GD: Set the path to libpng install prefix.

--with-xpm-dir=DIR

GD: Set the path to libXpm install prefix.

--with-freetype-dir=DIR

GD: Set the path to freetype2 install prefix.

--with-ttf[=DIR]

GD: Include FreeType 1.x support.

--with-t1lib[=DIR]

GD: Include T1lib support.

--with-cpdflib[=DIR]

Include cpdflib support (requires cpdflib >= 2). DIR is the cpdfllib install directory, defaults to /usr.

--with-jpeg-dir[=DIR]

jpeg dir for cpdflib 2.x.

--with-tiff-dir[=DIR]

tiff dir for cpdflib 2.x.

--with-pdflib[=DIR]

Include PDFlib support. DIR is the pdflib base install directory, defaults to /usr/local. Set DIR to shared to build as dl, or shared,DIR to build as dl and still specify DIR.

--with-jpeg-dir[=DIR]

PDFLIB: define libjpeg install directory.

--with-png-dir[=DIR]

PDFLIB: define libpng install directory.

--with-tiff-dir[=DIR]

PDFLIB: define libtiff install directory.

--with-swf[=DIR]

Include swf support.

--without-gd

Disable GD support. PHP 3 only!

--with-imagick

Include ImageMagick support. DIR is the install directory, and if left out, PHP will try to find it on its own. [experimental]. PHP 3 only!

--with-ming[=DIR]

Include ming support.

Misc options

--enable-force-cgi-redirect

Enable the security check for internal server redirects. You should use this if you are running the CGI version with Apache.

--enable-discard-path

If this is enabled, the PHP CGI binary can safely be placed outside of the web tree and people will not be able to circumvent .htaccess security.

--with-fastcgi=SRCDIR

Build PHP as FastCGI application.

--enable-debug

Compile with debugging symbols.

--with-layout=TYPE

Sets how installed files will be laid out. Type is one of PHP (default) or GNU.

--with-pear=DIR

Install PEAR in DIR (default PREFIX/lib/php).

--without-pear

Do not install PEAR.

--with-openssl[=DIR]

Include OpenSSL support (requires OpenSSL >= 0.9.5).

--enable-sigchild

Enable PHP's own SIGCHLD handler.

--disable-rpath

Disable passing additional runtime library search paths.

--enable-libgcc

Enable explicitly linking against libgcc.

--enable-dmalloc

Enable dmalloc.

--enable-php-streams

Include experimental php streams. Do not use unless you are testing the code!

--with-zlib-dir=<DIR>

Define the location of zlib install directory.

--with-zlib[=DIR]

Include zlib support (requires zlib >= 1.0.9). DIR is the zlib install directory.

--with-aspell[=DIR]

Include ASPELL support.

--enable-bcmath

Enable bc style precision math functions.

--with-bz2[=DIR]

Include BZip2 support.

--enable-calendar

Enable support for calendar conversion.

--with-ccvs[=DIR]

Include CCVS support.

--with-crack[=DIR]

Include crack support.

--enable-ctype

Enable ctype support.

--with-curl[=DIR]

Include CURL support.

--with-cybercash[=DIR]

Include CyberCash support. DIR is the CyberCash MCK install directory.

--with-cybermut[=DIR]

Include CyberMut (French Credit Mutuel telepaiement).

--with-cyrus

Include cyrus IMAP support.

--enable-exif

Enable exif support.

--with-fdftk[=DIR]

Include fdftk support.

--enable-ftp

Enable FTP support.

--with-gettext[=DIR]

Include GNU gettext support. DIR is the gettext install directory, defaults to /usr/local.

--with-gmp

Include gmp support.

--with-hyperwave

Include Hyperwave support.

--with-icap[=DIR]

Include ICAP support.

--with-iconv[=DIR]

Include iconv support.

--with-imap[=DIR]

Include IMAP support. DIR is the c-client install prefix.

--with-kerberos[=DIR]

IMAP: Include Kerberos support. DIR is the Kerberos install dir.

--with-imap-ssl[=DIR]

IMAP: Include SSL support. DIR is the OpenSSL install dir.

--with-ircg-config

Path to the ircg-config script.

--with-ircg

Include ircg support.

--with-java[=DIR]

Include Java support. DIR is the base install directory for the JDK. This extension can only be built as a shared dl.

--with-ldap[=DIR]

Include LDAP support. DIR is the LDAP base install directory.

--enable-mailparse

Enable mailparse support.

--enable-mbstring

Enable multibyte string support.

--enable-mbstr-enc-trans

Enable japanese encoding translation.

--with-mcal[=DIR]

Include MCAL support.

--with-mcrypt[=DIR]

Include mcrypt support. DIR is the mcrypt install directory.

--with-mhash[=DIR]

Include mhash support. DIR is the mhash install directory.

--with-mnogosearch[=DIR]

Include mnoGoSearch support. DIR is the mnoGoSearch base install directory, defaults to /usr/local/mnogosearch.

--with-muscat[=DIR]

Include muscat support.

--with-ncurses

Include ncurses support.

--enable-pcntl

Enable experimental pcntl support (CGI ONLY!)

--without-pcre-regex

Do not include Perl Compatible Regular Expressions support. Use --with-pcre-regex=DIR to specify DIR where PCRE's include and library files are located, if not using bundled library.

--with-pfpro[=DIR]

Include Verisign Payflow Pro support.

--disable-posix

Disable POSIX-like functions.

--with-pspell[=DIR]

Include PSPELL support.

--with-qtdom

Include QtDOM support (requires Qt >= 2.2.0).

--with-libedit[=DIR]

Include libedit readline replacement.

--with-readline[=DIR]

Include readline support. DIR is the readline install directory.

--with-recode[=DIR]

Include recode support. DIR is the recode install directory.

--with-satellite[=DIR]

Enable CORBA support via Satellite (EXPERIMENTAL) DIR is the base directory for ORBit.

--with-mm[=DIR]

Include mm support for session storage.

--enable-trans-sid

Enable transparent session id propagation.

--disable-session

Disable session support.

--enable-shmop

Enable shmop support.

--with-snmp[=DIR]

Include SNMP support. DIR is the SNMP base install directory, defaults to searching through a number of common locations for the snmp install. Set DIR to shared to build as a dl, or shared,DIR to build as a dl and still specify DIR.

--enable-ucd-snmp-hack

Enable UCD SNMP hack.

--enable-sockets

Enable sockets support.

--with-regex=TYPE

regex library type: system, apache, php.

--with-system-regex

Use system regex library (deprecated).

--enable-sysvsem

Enable System V semaphore support.

--enable-sysvshm

Enable the System V shared memory support.

--with-vpopmail[=DIR]

Include vpopmail support.

--with-tsrm-pthreads

Use POSIX threads (default).

--enable-shared[=PKGS]

Build shared libraries [default=yes].

--enable-static[=PKGS]

Build static libraries [default=yes].

--enable-fast-install[=PKGS]

Optimize for fast installation [default=yes].

--with-gnu-ld

Assume the C compiler uses GNU ld [default=no].

--disable-libtool-lock

Avoid locking (might break parallel builds).

--with-pic

Try to use only PIC/non-PIC objects [default=use both].

--with-yaz[=DIR]

Include YAZ support (ANSI/NISO Z39.50). DIR is the YAZ bin install directory.

--enable-memory-limit

Compile with memory limit support.

--disable-url-fopen-wrapper

Disable the URL-aware fopen wrapper that allows accessing files via HTTP or FTP.

--enable-versioning

Export only required symbols. See INSTALL for more information.

--disable-bcmath

Compile without bc style precision math functions. PHP 3 only!

--with-imsp[=DIR]

Include IMSp support (DIR is IMSP's include dir and libimsp.a dir). PHP 3 only!

--with-ftp

Include FTP support. PHP 3 only!

--with-mck[=DIR]

Include Cybercash MCK support. DIR is the cybercash mck build directory, defaults to /usr/src/mck-3.2.0.3-linux for help look in extra/cyberlib. PHP 3 only!

--disable-overload

Disable user-space object overloading support. PHP 3 only!

--enable-yp

Include YP support. PHP 3 only!

--with-zip

Include ZIP support (requires zziplib >= 0.10.6). PHP 3 only!

--with-mod-dav=DIR

Include DAV support through Apache's mod_dav, DIR is mod_dav's installation directory (Apache module version only!) PHP 3 only!

--enable-debugger

Compile with remote debugging functions. PHP 3 only!

--enable-versioning

Take advantage of versioning and scoping provided by Solaris 2.x and Linux. PHP 3 only!

PHP options

--enable-maintainer-mode

Enable make rules and dependencies not useful (and sometimes confusing) to the casual installer.

--with-config-file-path=PATH

Sets the path in which to look for `php.ini`, defaults to PREFIX/lib.

--enable-safe-mode

Enable safe mode by default.

--with-exec-dir[=DIR]

Only allow executables in DIR when in safe mode defaults to /usr/local/php/bin.

--enable-magic-quotes

Enable magic quotes by default.

--disable-short-tags

Disable the short-form <? start tag by default.

Server options

--with-aolserver=DIR

Specify path to the installed AOLserver.

--with-apxs[=FILE]

Build shared Apache module. FILE is the optional pathname to the Apache apxs tool; defaults to apxs.

--with-apache[=DIR]

Build Apache module. DIR is the top-level Apache build directory, defaults to /usr/local/apache.

--with-mod_charset

Enable transfer tables for mod_charset (Rus Apache).

--with-apxs2[=FILE]

Build shared Apache 2.0 module. FILE is the optional pathname to the Apache apxs tool; defaults to apxs.

--with-fhttpd[=DIR]

Build fhttpd module. DIR is the fhttpd sources directory, defaults to /usr/local/src/fhttpd.

--with-isapi=DIR

Build PHP as an ISAPI module for use with Zeus.

--with-nsapi=DIR

Specify path to the installed Netscape Server.

--with-phttpd=DIR

No information yet.

--with-pi3web=DIR

Build PHP as a module for use with Pi3Web.

--with-roxen=DIR

Build PHP as a Pike module. DIR is the base Roxen directory, normally /usr/local/roxen/server.

--enable-roxen-zts

Build the Roxen module using Zend Thread Safety.

--with-servlet[=DIR]

Include servlet support. DIR is the base install directory for the JSDK. This SAPI prereqs the java extension must be built as a shared dl.

--with-thttpd=SRCDIR

Build PHP as thttpd module.

--with-tux=MODULEDIR

Build PHP as a TUX module (Linux only).

XML options

--with-dom[=DIR]

Include DOM support (requires libxml >= 2.4.2). DIR is the libxml install directory, defaults to /usr.

--disable-xml

Disable XML support using bundled expat lib.

--with-expat-dir=DIR

XML: external libexpat install dir.

--with-xmlrpc[=DIR]

Include XMLRPC-EPI support.

--enable-wddx

Enable WDDX support.

Capitolo 3. Configuration

The configuration file

The configuration file (called `php3.ini` in PHP 3.0, and simply `php.ini` as of PHP 4.0) is read when PHP starts up. For the server module versions of PHP, this happens only once when the web server is started. For the CGI version, it happens on every invocation.

Esempio 3-1. `php.ini` example

```
; any text on a line after an unquoted semicolon (;) is ignored
[php] ; section markers (text within square brackets) are also ignored
; Boolean values can be set to either:
;    true, on, yes
; or false, off, no, none
register_globals = off
magic_quotes_gpc = yes

; you can enclose strings in double-quotes
include_path = ".:/usr/local/lib/php"

; backslashes are treated the same as any other character
include_path = ".;c:\php\lib"
```

When using PHP as an Apache module, you can also change the configuration settings using directives in Apache configuration files and .htaccess files (You will need "AllowOverride Options" or "AllowOverride All" privileges)

With PHP 3.0, there are Apache directives that correspond to each configuration setting in the `php3.ini` name, except the name is prefixed by "php3_".

With PHP 4.0, there are several Apache directives that allow you to change the PHP configuration from within the Apache configuration file itself.

php_value *`name value`*

This sets the value of the specified variable.

php_flag *`name on|off`*

This is used to set a Boolean configuration option.

php_admin_value *`name value`*

This sets the value of the specified variable. "Admin" configuration settings can only be set from within the main Apache configuration files, and not from .htaccess files.

php_admin_flag *`name on|off`*

This is used to set a Boolean configuration option.

Esempio 3-2. Apache configuration example

```
<IfModule mod_php4.c>
  php_value include_path ".:/usr/local/lib/php"
  php_flag safe_mode on
</IfModule>
<IfModule mod_php3.c>
  php3_include_path ".:/usr/local/lib/php"
  php3_safe_mode on
</IfModule>
```

You can view the settings of the configuration values in the output of phpinfo(). You can also access the values of individual configuration settings using get_cfg_var().

General Configuration Directives

`allow_url_fopen` boolean

This option enables the URL-aware fopen wrappers that enable accessing URL object like files. Default wrappers are provided for the access of remote files using the ftp or http protocol, some extensions like zlib may register additional wrappers.

> **Nota:** This option was introduced immediately after the release of version 4.0.3. For versions up to and including 4.0.3 you can only disable this feature at compile time by using the configuration switch `--disable-url-fopen-wrapper`.

`asp_tags` boolean

Enables the use of ASP-like <% %> tags in addition to the usual <?php ?> tags. This includes the variable-value printing shorthand of <%= $value %>. For more information, see Escaping from HTML.

> **Nota:** Support for ASP-style tags was added in 3.0.4.

`auto_append_file` string

Specifies the name of a file that is automatically parsed after the main file. The file is included as if it was called with the include() function, so include_path is used.

The special value none disables auto-appending.

> **Nota:** If the script is terminated with exit(), auto-append will *not* occur.

`auto_prepend_file` string

Specifies the name of a file that is automatically parsed before the main file. The file is included as if it was called with the include() function, so include_path is used.

The special value none disables auto-prepending.

`cgi_ext` string

`display_errors` boolean

This determines whether errors should be printed to the screen as part of the HTML output or not.

`doc_root` string

PHP's "root directory" on the server. Only used if non-empty. If PHP is configured with safe mode, no files outside this directory are served.

`engine` boolean

This directive is really only useful in the Apache module version of PHP. It is used by sites that would like to turn PHP parsing on and off on a per-directory or per-virtual server basis. By putting **`engine off`** in the appropriate places in the `httpd.conf` file, PHP can be enabled or disabled.

`error_log` string

Name of file where script errors should be logged. If the special value `syslog` is used, the errors are sent to the system logger instead. On UNIX, this means syslog(3) and on Windows NT it means the event log. The system logger is not supported on Windows 95.

`error_reporting` integer

Set the error reporting level. The parameter is an integer representing a bit field. Add the values of the error reporting levels you want.

Tabella 3-1. Error Reporting Levels

bit value	enabled reporting
1	normal errors
2	normal warnings
4	parser errors
8	non-critical style-related warnings

The default value for this directive is 7 (normal errors, normal warnings and parser errors are shown).

`html_errors` boolean

Turn off HTML tags in error messages.

open_basedir string

Limit the files that can be opened by PHP to the specified directory-tree.

When a script tries to open a file with, for example, fopen or gzopen, the location of the file is checked. When the file is outside the specified directory-tree, PHP will refuse to open it. All symbolic links are resolved, so it's not possible to avoid this restriction with a symlink.

The special value . indicates that the directory in which the script is stored will be used as base-directory.

Under Windows, separate the directories with a semicolon. On all other systems, separate the directories with a colon. As an Apache module, open_basedir paths from parent directories are now automatically inherited.

The restriction specified with open_basedir is actually a prefix, not a directory name. This means that "open_basedir = /dir/incl" also allows access to "/dir/include" and "/dir/incls" if they exist. When you want to restrict access to only the specified directory, end with a slash. For example: "open_basedir = /dir/incl/"

> **Nota:** Support for multiple directories was added in 3.0.7.

The default is to allow all files to be opened.

gpc_order string

Set the order of GET/POST/COOKIE variable parsing. The default setting of this directive is "GPC". Setting this to "GP", for example, will cause PHP to completely ignore cookies and to overwrite any GET method variables with POST-method variables of the same name.

Note, that this option is not available in PHP 4. Use variables_order instead.

variables_order string

Set the order of the EGPCS (Environment, GET, POST, Cookie, Server) variable parsing. The default setting of this directive is "EGPCS". Setting this to "GP", for example, will cause PHP to completely ignore environment variables, cookies and server variables, and to overwrite any GET method variables with POST-method variables of the same name.

See also register_globals.

ignore_user_abort string

On by default. If changed to Off scripts will be terminated as soon as they try to output something after a client has aborted their connection. ignore_user_abort().

implicit_flush boolean

`FALSE` by default. Changing this to `TRUE` tells PHP to tell the output layer to flush itself automatically after every output block. This is equivalent to calling the `PHP` function flush() after each and every call to print() or echo() and each and every `HTML` block.

When using `PHP` within an web environment, turning this option on has serious performance implications and is generally recommended for debugging purposes only. This value defaults to `TRUE` when operating under the `CLI SAPI`.

`include_path` string

Specifies a list of directories where the require(), include() and **fopen_with_path()** functions look for files. The format is like the system's PATH environment variable: a list of directories separated with a colon in UNIX or semicolon in Windows.

Esempio 3-3. UNIX include_path

```
include_path=.:/home/httpd/php-lib
```

Esempio 3-4. Windows include_path

```
include_path=".;c:\www\phplib"
```

The default value for this directive is `.` (only the current directory).

`isapi_ext` string

`log_errors` boolean

Tells whether script error messages should be logged to the server's error log. This option is thus server-specific.

`magic_quotes_gpc` boolean

Sets the magic_quotes state for GPC (Get/Post/Cookie) operations. When magic_quotes are on, all ' (single-quote), " (double quote), \ (backslash) and NUL's are escaped with a backslash automatically. If magic_quotes_sybase is also on, a single-quote is escaped with a single-quote instead of a backslash.

`magic_quotes_runtime` boolean

If `magic_quotes_runtime` is enabled, most functions that return data from any sort of external source including databases and text files will have quotes escaped with a backslash. If `magic_quotes_sybase` is also on, a single-quote is escaped with a single-quote instead of a backslash.

`magic_quotes_sybase` boolean

If `magic_quotes_sybase` is also on, a single-quote is escaped with a single-quote instead of a backslash if `magic_quotes_gpc` or `magic_quotes_runtime` is enabled.

`max_execution_time` integer

This sets the maximum time in seconds a script is allowed to run before it is terminated by the parser. This helps prevent poorly written scripts from tying up the server. The default setting is `30`.

The maximum execution time is not affected by system calls, the sleep() function, etc. Please see the set_time_limit() function for more details.

memory_limit integer

This sets the maximum amount of memory in bytes that a script is allowed to allocate. This helps prevent poorly written scripts for eating up all available memory on a server.

nsapi_ext string

precision integer

The number of significant digits displayed in floating point numbers.

register_argc_argv boolean

Tells PHP whether to declare the argv & argc variables (that would contain the GET information).

See also command line. Also, this directive became available in PHP 4.0.0 and was always "on" before that.

post_max_size integer

Sets max size of post data allowed. This setting also affects file upload. To upload large files, this value must be larger than upload_max_filesize.

If memory limit is enabled by configure script, memory_limit also affects file uploading. Generally speaking, memory_limit should be larger than post_max_size.

register_globals boolean

Tells whether or not to register the EGPCS (Environment, GET, POST, Cookie, Server) variables as global variables. You may want to turn this off if you don't want to clutter your scripts' global scope with user data. This makes the most sense when coupled with track_vars - in which case you can access all of the EGPCS variables through the `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, and `$_SERVER` arrays in the global scope.

Please note that you need to set AllowOveride All in your Directory block in the apache config file for this to work.

short_open_tag boolean

Tells whether the short form (**`<? ?>`**) of PHP's open tag should be allowed. If you want to use PHP in combination with XML, you have to disable this option. If disabled, you must use the long form of the open tag (**`<?php ?>`**).

sql.safe_mode boolean

track_errors boolean

If enabled, the last error message will always be present in the global variable `$php_errormsg`.

`track_vars` boolean

If enabled, then Environment, GET, POST, Cookie, and Server variables can be found in the global associative arrays `$_ENV`, `$_GET`, `$_POST`, `$_COOKIE`, and `$_SERVER`.

Note that as of PHP 4.0.3, track_vars is always turned on.

`upload_tmp_dir` string

The temporary directory used for storing files when doing file upload. Must be writable by whatever user PHP is running as.

`upload_max_filesize` integer

The maximum size of an uploaded file. The value is in bytes.

`user_dir` string

The base name of the directory used on a user's home directory for PHP files, for example `public_html`.

`warn_plus_overloading` boolean

If enabled, this option makes PHP output a warning when the plus (`+`) operator is used on strings. This is to make it easier to find scripts that need to be rewritten to using the string concatenator instead (`.`).

Safe Mode Configuration Directives

`safe_mode` boolean

Whether to enable PHP's safe mode. Read the Security and Safe Mode chapters for more information.

`safe_mode_gid` boolean

Whether to use `UID` (`Off`) or `GID` (`On`) checking upon file access. See Safe Mode for more information.

`safe_mode_exec_dir` string

If PHP is used in safe mode, system() and the other functions executing system programs refuse to start programs that are not in this directory.

`safe_mode_include_dir` string

`UID`/`GID` checks are bypassed when including files from this directory and its subdirectories (directory must also be in include_path or full path must including).

Debugger Configuration Directives

`debugger.host` string

DNS name or IP address of host used by the debugger.

debugger.port string

Port number used by the debugger.

debugger.enabled boolean

Whether the debugger is enabled.

Extension Loading Directives

enable_dl boolean

This directive is really only useful in the Apache module version of PHP. You can turn dynamic loading of PHP extensions with dl() on and off per virtual server or per directory.

The main reason for turning dynamic loading off is security. With dynamic loading, it's possible to ignore all the safe_mode and open_basedir restrictions.

The default is to allow dynamic loading, except when using safe-mode. In safe-mode, it's always imposible to use dl().

extension_dir string

In what directory PHP should look for dynamically loadable extensions.

extension string

Which dynamically loadable extensions to load when PHP starts up.

mSQL Configuration Directives

msql.allow_persistent boolean

Whether to allow persistent mSQL connections.

msql.max_persistent integer

The maximum number of persistent mSQL connections per process.

msql.max_links integer

The maximum number of mSQL connections per process, including persistent connections.

Postgres Configuration Directives

pgsql.allow_persistent boolean

Whether to allow persistent Postgres connections.

pgsql.max_persistent integer

The maximum number of persistent Postgres connections per process.

pgsql.max_links integer

The maximum number of Postgres connections per process, including persistent connections.

SESAM Configuration Directives

sesam_oml string

Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. The BS2000 PLAM library must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

sesam_configfile string

Name of SESAM application configuration file. Required for using SESAM functions. The BS2000 file must be readable by the apache server's user id.

The application configuration file will usually contain a configuration like (see SESAM reference manual):

```
CNF=B
NAM=K
NOTYPE
```

sesam_messagecatalog string

Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive.

The message catalog must be set ACCESS=READ,SHARE=YES because it must be readable by the apache server's user id.

Sybase Configuration Directives

sybase.allow_persistent boolean

Whether to allow persistent Sybase connections.

sybase.max_persistent integer

The maximum number of persistent Sybase connections per process.

sybase.max_links integer

The maximum number of Sybase connections per process, including persistent connections.

Sybase-CT Configuration Directives

sybct.allow_persistent boolean

Whether to allow persistent Sybase-CT connections. The default is on.

sybct.max_persistent integer

The maximum number of persistent Sybase-CT connections per process. The default is -1 meaning unlimited.

sybct.max_links integer

The maximum number of Sybase-CT connections per process, including persistent connections. The default is -1 meaning unlimited.

sybct.min_server_severity integer

Server messages with severity greater than or equal to sybct.min_server_severity will be reported as warnings. This value can also be set from a script by calling sybase_min_server_severity(). The default is 10 which reports errors of information severity or greater.

sybct.min_client_severity integer

Client library messages with severity greater than or equal to sybct.min_client_severity will be reported as warnings. This value can also be set from a script by calling sybase_min_client_severity(). The default is 10 which effectively disables reporting.

sybct.login_timeout integer

The maximum time in seconds to wait for a connection attempt to succeed before returning failure. Note that if max_execution_time has been exceeded when a connection attempt times out, your script will be terminated before it can take action on failure. The default is one minute.

sybct.timeout integer

The maximum time in seconds to wait for a select_db or query operation to succeed before returning failure. Note that if max_execution_time has been exceeded when am operation times out, your script will be terminated before it can take action on failure. The default is no limit.

sybct.hostname string

The name of the host you claim to be connecting from, for display by sp_who. The default is none.

Informix Configuration Directives

ifx.allow_persistent boolean

Whether to allow persistent Informix connections.

ifx.max_persistent integer

The maximum number of persistent Informix connections per process.

ifx.max_links integer

The maximum number of Informix connections per process, including persistent connections.

ifx.default_host string

The default host to connect to when no host is specified in ifx_connect() or ifx_pconnect().

ifx.default_user string

The default user id to use when none is specified in ifx_connect() or ifx_pconnect().

ifx.default_password string

The default password to use when none is specified in ifx_connect() or ifx_pconnect().

ifx.blobinfile boolean

Set to `TRUE` if you want to return blob columns in a file, `FALSE` if you want them in memory. You can override the setting at runtime with ifx_blobinfile_mode().

ifx.textasvarchar boolean

Set to `TRUE` if you want to return TEXT columns as normal strings in select statements, `FALSE` if you want to use blob id parameters. You can override the setting at runtime with ifx_textasvarchar().

ifx.byteasvarchar boolean

Set to `TRUE` if you want to return BYTE columns as normal strings in select queries, `FALSE` if you want to use blob id parameters. You can override the setting at runtime with ifx_textasvarchar().

ifx.charasvarchar boolean

Set to `TRUE` if you want to trim trailing spaces from CHAR columns when fetching them.

ifx.nullformat boolean

Set to `TRUE` if you want to return `NULL` columns as the literal string `"NULL"`, `FALSE` if you want them returned as the empty string "". You can override this setting at runtime with ifx_nullformat().

BC Math Configuration Directives

bcmath.scale integer

Number of decimal digits for all bcmath functions.

Browser Capability Configuration Directives

browscap string

Name of browser capabilities file. See also get_browser().

Multi-Byte String Configuration Directives

mbstring.internal_encoding string

`mbstring.internal_encoding` defines default internal character encoding.

mbstring.http_input string

`mbstring.http_input` defines default HTTP input character encoding.

mbstring.http_output string

`mbstring.http_output` defines default HTTP output character encoding.

mbstring.detect_order string

`mbstring.detect_order` defines default character encoding detection order.

mbstring.substitute_character string

`mbstring.substitute_character` defines character to substitute for invalid character codes.

Exif Configuration Directives

Exif supports automatically conversion for Unicode and JIS character encodings of user comments when module mbstring is available. This is done by first decoding the comment using the specified characterset. The result is then encoded with another characterset which should match your http output.

exif.encode_unicode string

`exif.encode_unicode` defines the characterset UNICODE user comments are handled. This defaults to ISO-8859-15 which should work for most non asian counties. The setting can be empty or must be an encoding supported by mbstring. If it is empty the current internal encoding of mbstring is used.

exif.decode_unicode_motorola string

`exif.decode_unicode_motorola` defines the image internal characterset for Unicode encoded user comments if image is in motorola byte order (big-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is UCS-2BE.

exif.decode_unicode_intel string

`exif.decode_unicode_intel` defines the image internal characterset for Unicode encoded user comments if image is in intel byte order (little-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is UCS-2LE.

exif.encode_jis string

`exif.encode_jis` defines the characterset JIS user comments are handled. This defaults to an empty value which forces the functions to use the current internal encoding of mbstring.

`exif.decode_jis_motorola` string

`exif.decode_jis_motorola` defines the image internal characterset for JIS encoded user comments if image is in motorola byte order (big-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is JIS.

`exif.decode_jis_intel` string

`exif.decode_jis_intel` defines the image internal characterset for JIS encoded user comments if image is in intel byte order (little-endian). This setting cannot be empty but you can specify a list of encodings supported by mbstring. The default is JIS.

Capitolo 4. Security

PHP is a powerful language and the interpreter, whether included in a web server as a module or executed as a separate CGI binary, is able to access files, execute commands and open network connections on the server. These properties make anything run on a web server insecure by default. PHP is designed specifically to be a more secure language for writing CGI programs than Perl or C, and with correct selection of compile-time and runtime configuration options, and proper coding practices, it can give you exactly the combination of freedom and security you need.

As there are many different ways of utilizing PHP, there are many configuration options controlling its behaviour. A large selection of options guarantees you can use PHP for a lot of purposes, but it also means there are combinations of these options and server configurations that result in an insecure setup.

The configuration flexibility of PHP is equally rivalled by the code flexibility. PHP can be used to build complete server applications, with all the power of a shell user, or it can be used for simple server-side includes with little risk in a tightly controlled environment. How you build that environment, and how secure it is, is largely up to the PHP developer.

This chapter starts with some general security advice, explains the different configuration option combinations and the situations they can be safely used, and describes different considerations in coding for different levels of security.

General considerations

A completely secure system is a virtual impossibility, so an approach often used in the security profession is one of balancing risk and usability. If every variable submitted by a user required two forms of biometric validation (such as a retinal scan and a fingerprint), you would have an extremely high level of accountability. It would also take half an hour to fill out a fairly complex form, which would tend to encourage users to find ways of bypassing the security.

The best security is often inobtrusive enough to suit the requirements without the user being prevented from accomplishing their work, or over-burdening the code author with excessive complexity. Indeed, some security attacks are merely exploits of this kind of overly built security, which tends to erode over time.

A phrase worth remembering: A system is only as good as the weakest link in a chain. If all transactions are heavily logged based on time, location, transaction type, etc. but the user is only verified based on a single cookie, the validity of tying the users to the transaction log is severely weakened.

When testing, keep in mind that you will not be able to test all possibilities for even the simplest of pages. The input you may expect will be completely unrelated to the input given by a disgruntled employee, a cracker with months of time on their hands, or a housecat walking across the keyboard. This is why it’s best to look at the code from a logical perspective, to discern where unexpected data can be introduced, and then follow how it is modified, reduced, or amplified.

The Internet is filled with people trying to make a name for themselves by breaking your code, crashing your site, posting inappropriate content, and otherwise making your day interesting. It doesn’t matter if you have a small or large site, you are a target by simply being online, by having a server that can be connected to. Many cracking programs do not discern by size, they simply trawl massive IP blocks looking for victims. Try not to become one.

Installed as CGI binary

Possible attacks

Using PHP as a CGI binary is an option for setups that for some reason do not wish to integrate PHP as a module into server software (like Apache), or will use PHP with different kinds of CGI wrappers to create safe chroot and setuid environments for scripts. This setup usually involves installing executable PHP binary to the web server cgi-bin directory. CERT advisory CA-96.11 (http://www.cert.org/advisories/CA-96.11.interpreters_in_cgi_bin_dir.html) recommends against placing any interpreters into cgi-bin. Even if the PHP binary can be used as a standalone interpreter, PHP is designed to prevent the attacks this setup makes possible:

- Accessing system files: `http://my.host/cgi-bin/php?/etc/passwd`

  The query information in a url after the question mark (?) is passed as command line arguments to the interpreter by the CGI interface. Usually interpreters open and execute the file specified as the first argument on the command line.

  When invoked as a CGI binary, PHP refuses to interpret the command line arguments.

- Accessing any web document on server: `http://my.host/cgi-bin/php/secret/doc.html`

  The path information part of the url after the PHP binary name, `/secret/doc.html` is conventionally used to specify the name of the file to be opened and interpreted by the CGI program. Usually some web server configuration directives (Apache: Action) are used to redirect requests to documents like `http://my.host/secret/script.php` to the PHP interpreter. With this setup, the web server first checks the access permissions to the directory `/secret`, and after that creates the redirected request `http://my.host/cgi-bin/php/secret/script.php`. Unfortunately, if the request is originally given in this form, no access checks are made by web server for file `/secret/script.php`, but only for the `/cgi-bin/php` file. This way any user able to access `/cgi-bin/php` is able to access any protected document on the web server.

  In PHP, compile-time configuration option --enable-force-cgi-redirect and runtime configuration directives doc_root and user_dir can be used to prevent this attack, if the server document tree has any directories with access restrictions. See below for full the explanation of the different combinations.

Case 1: only public files served

If your server does not have any content that is not restricted by password or ip based access control, there is no need for these configuration options. If your web server does not allow you to do redirects, or the server does not have a way to communicate to the PHP binary that the request is a safely redirected request, you can specify the option --enable-force-cgi-redirect to the configure script. You still have to make sure your PHP scripts do not rely on one or another way of calling the script, neither by directly `http://my.host/cgi-bin/php/dir/script.php` nor by redirection `http://my.host/dir/script.php`.

Redirection can be configured in Apache by using AddHandler and Action directives (see below).

Case 2: using --enable-force-cgi-redirect

This compile-time option prevents anyone from calling PHP directly with a url like `http://my.host/cgi-bin/php/secretdir/script.php`. Instead, PHP will only parse in this mode if it has gone through a web server redirect rule.

Usually the redirection in the Apache configuration is done with the following directives:

```
Action php-script /cgi-bin/php
AddHandler php-script .php
```

This option has only been tested with the Apache web server, and relies on Apache to set the non-standard CGI environment variable REDIRECT_STATUS on redirected requests. If your web server does not support any way of telling if the request is direct or redirected, you cannot use this option and you must use one of the other ways of running the CGI version documented here.

Case 3: setting doc_root or user_dir

To include active content, like scripts and executables, in the web server document directories is sometimes consider an insecure practice. If, because of some configuration mistake, the scripts are not executed but displayed as regular HTML documents, this may result in leakage of intellectual property or security information like passwords. Therefore many sysadmins will prefer setting up another directory structure for scripts that are accessible only through the PHP CGI, and therefore always interpreted and not displayed as such.

Also if the method for making sure the requests are not redirected, as described in the previous section, is not available, it is necessary to set up a script doc_root that is different from web document root.

You can set the PHP script document root by the configuration directive doc_root in the configuration file, or you can set the environment variable PHP_DOCUMENT_ROOT. If it is set, the CGI version of PHP will always construct the file name to open with this *`doc_root`* and the path information in the request, so you can be sure no script is executed outside this directory (except for *`user_dir`* below).

Another option usable here is user_dir. When user_dir is unset, only thing controlling the opened file name is *`doc_root`*. Opening an url like `http://my.host/~user/doc.php` does not result in opening a file under users home directory, but a file called `~user/doc.php` under doc_root (yes, a directory name starting with a tilde [~]).

If user_dir is set to for example `public_php`, a request like `http://my.host/~user/doc.php` will open a file called `doc.php` under the directory named `public_php` under the home directory of the user. If the home of the user is `/home/user`, the file executed is `/home/user/public_php/doc.php`.

`user_dir` expansion happens regardless of the *`doc_root`* setting, so you can control the document root and user directory access separately.

Case 4: PHP parser outside of web tree

A very secure option is to put the PHP parser binary somewhere outside of the web tree of files. In `/usr/local/bin`, for example. The only real downside to this option is that you will now have to

put a line similar to:

```
#!/usr/local/bin/php
```

as the first line of any file containing PHP tags. You will also need to make the file executable. That is, treat it exactly as you would treat any other CGI script written in Perl or sh or any other common scripting language which uses the `#!` shell-escape mechanism for launching itself.

To get PHP to handle PATH_INFO and PATH_TRANSLATED information correctly with this setup, the php parser should be compiled with the --enable-discard-path configure option.

Installed as an Apache module

When PHP is used as an Apache module it inherits Apache's user permissions (typically those of the "nobody" user). This has several impacts on security and authorization. For example, if you are using PHP to access a database, unless that database has built-in access control, you will have to make the database accessable to the "nobody" user. This means a malicious script could access and modify the database, even without a username and password. It's entirely possible that a web spider could stumble across a database administrator's web page, and drop all of your databases. You can protect against this with Apache authorization, or you can design your own access model using LDAP, .htaccess files, etc. and include that code as part of your PHP scripts.

Often, once security is established to the point where the PHP user (in this case, the apache user) has very little risk attached to it, it is discovered that PHP is now prevented from writing any files to user directories. Or perhaps it has been prevented from accessing or changing databases. It has equally been secured from writing good and bad files, or entering good and bad database transactions.

A frequent security mistake made at this point is to allow apache root permissions, or to escalate apache's abilitites in some other way.

Escalating the Apache user's permissions to root is extremely dangerous and may compromise the entire system, so sudo'ing, chroot'ing, or otherwise running as root should not be considered by those who are not security professionals.

There are some simpler solutions. By using open_basedir you can control and restrict what directories are allowed to be used for PHP. You can also set up apache-only areas, to restrict all web based activity to non-user, or non-system, files.

Filesystem Security

PHP is subject to the security built into most server systems with respect to permissions on a file and directory basis. This allows you to control which files in the filesystem may be read. Care should be taken with any files which are world readable to ensure that they are safe for reading by all users who have access to that filesystem.

Since PHP was designed to allow user level access to the filesystem, it's entirely possible to write a PHP script that will allow you to read system files such as /etc/passwd, modify your ethernet connections, send massive printer jobs out, etc. This has some obvious implications, in that you need to ensure that the files that you read from and write to are the appropriate ones.

Consider the following script, where a user indicates that they'd like to delete a file in their home directory. This assumes a situation where a PHP web interface is regularly used for file management, so the Apache user is allowed to delete files in the user home directories.

Esempio 4-1. Poor variable checking leads to....

```
<?php
// remove a file from the user's home directory
$username = $HTTP_POST_VARS['user_submitted_name'];
$homedir = "/home/$username";
$file_to_delete = "$userfile";
unlink ($homedir/$userfile);
echo "$file_to_delete has been deleted!";
?>
```

Since the username is postable from a user form, they can submit a username and file belonging to someone else, and delete files. In this case, you'd want to use some other form of authentication. Consider what could happen if the variables submitted were "../etc/" and "passwd". The code would then effectively read:

Esempio 4-2. ... A filesystem attack

```
<?php
// removes a file from anywhere on the hard drive that
// the PHP user has access to. If PHP has root access:
$username = "../etc/";
$homedir = "/home/../etc/";
$file_to_delete = "passwd";
unlink ("/home/../etc/passwd");
echo "/home/../etc/passwd has been deleted!";
?>
```

There are two important measures you should take to prevent these issues.

- Only allow limited permissions to the PHP web user binary.
- Check all variables which are submitted.

Here is an improved script:

Esempio 4-3. More secure file name checking

```
<?php
// removes a file from the hard drive that
// the PHP user has access to.
$username = $HTTP_SERVER_VARS['REMOTE_USER']; // using an authentication mechanisim
```

```
$homedir = "/home/$username";

$file_to_delete = basename("$userfile"); // strip paths
unlink ($homedir/$file_to_delete);

$fp = fopen("/home/logging/filedelete.log","+a"); //log the deletion
$logstring = "$username $homedir $file_to_delete";
fputs ($fp, $logstring);
fclose($fp);

echo "$file_to_delete has been deleted!";
?>
```

However, even this is not without it’s flaws. If your authentication system allowed users to create their own user logins, and a user chose the login "../etc/", the system is once again exposed. For this reason, you may prefer to write a more customized check:

Esempio 4-4. More secure file name checking

```
<?php
$username = $HTTP_SERVER_VARS['REMOTE_USER']; // using an authentication mechanisim
$homedir = "/home/$username";

if (!ereg('^[^./][^/]*$', $userfile))
     die('bad filename'); //die, do not process

if (!ereg('^[^./][^/]*$', $username))
     die('bad username'); //die, do not process
//etc...
?>
```

Depending on your operating system, there are a wide variety of files which you should be concerned about, including device entries (/dev/ or COM1), configuration files (/etc/ files and the .ini files), well known file storage areas (/home/, My Documents), etc. For this reason, it’s usually easier to create a policy where you forbid everything except for what you explicitly allow.

Database Security

Nowadays, databases are cardinal components of any web based application by enabling websites to provide varying dynamic content. Since very sensitive or secret informations can be stored in such database, you should strongly consider to protect them somehow.

To retrieve or to store any information you need to connect to the database, send a legitimate query, fetch the result, and close the connecion. Nowadays, the commonly used query language in this interaction is the Structured Query Language (SQL). See how an attacker can tamper with an SQL query.

As you can realize, PHP cannot protect your database by itself. The following sections aim to be an introduction into the very basics of how to access and manipulate databases within PHP scripts.

Keep in mind this simple rule: defence in depth. In the more place you take the more action to increase the protection of your database, the less probability of that an attacker succeeds, and exposes or abuse any stored secret information. Good design of the database schema and the application deals with your greatest fears.

Designing Databases

The first step is always to create the database, unless you want to use an existing third party's one. When a database is created, it is assigned to an owner, who executed the creation statement. Usually, only the owner (or a superuser) can do anything with the objects in that database, and in order to allow other users to use it, privileges must be granted.

Applications should never connect to the database as its owner or a superuser, because these users can execute any query at will, for example, modifying the schema (e.g. dropping tables) or deleting its entire content.

You may create different database users for every aspect of your application with very limited rights to database objects. The most required privileges should be granted only, and avoid that the same user can interact with the database in different use cases. This means that if intruders gain access to your database using one of these credentials, they can only effect as many changes as your application can.

You are encouraged not to implement all the business logic in the web application (i.e. your script), instead to do it in the database schema using views, triggers or rules. If the system evolves, new ports will be intended to open to the database, and you have to reimplement the logic in each separate database client. Over and above, triggers can be used to transparently and automatically handle fields, which often provides insight when debugging problems with your application or tracing back transactions.

Connecting to Database

You may want to estabilish the connections over SSL to encrypt client/server communications for increased security, or you can use ssh to encrypt the network connection between clients and the database server. If either of them is done, then monitoring your traffic and gaining informations in this way will be a hard work.

Encrypted Storage Model

SSL/SSH protects data travelling from the client to the server, SSL/SSH does not protect the persistent data stored in a database. SSL is an on-the-wire protocol.

Once an attacker gains access to your database directly (bypassing the webserver), the stored sensitive data may be exposed or misused, unless the information is protected by the database itself. Encrypting the data is a good way to mitigate this threat, but very few databases offer this type of data encryption.

The easiest way to work around this problem is to first create your own encryption package, and then use it from within your PHP scripts. PHP can assist you in this case with its several extensions, such as Mcrypt and Mhash, covering a wide variety of encryption algorithms. The script encrypts the data

be stored first, and decrypts it when retrieving. See the references for further examples how encryption works.

In case of truly hidden data, if its raw representation is not needed (i.e. not be displayed), hashing may be also taken into consideration. The well-known example for the hashing is storing the MD5 hash of a password in a database, instead of the password itself. See also crypt() and md5().

Esempio 4-5. Using hashed password field

```
// storing password hash
$query  = sprintf("INSERT INTO users(name,pwd) VALUES('%s','%s');",
            addslashes($username), md5($password));
$result = pg_exec($connection, $query);

// querying if user submitted the right password
$query = sprintf("SELECT 1 FROM users WHERE name='%s' AND pwd='%s';",
            addslashes($username), md5($password));
$result = pg_exec($connection, $query);

if (pg_numrows($result) > 0) {
    echo "Welcome, $username!";
}
else {
    echo "Authentication failed for $username.";
}
```

SQL Injection

Many web developers are unaware of how SQL queries can be tampered with, and assume that an SQL query is a trusted command. It means that SQL queries are able to circumvent access controls, thereby bypassing standard authentication and authorization checks, and sometimes SQL queries even may allow access to host operating system level commands.

Direct SQL Command Injection is a technique where an attacker creates or alters existing SQL commands to expose hidden data, or to override valuable ones, or even to execute dangerous system level commands on the database host. This is accomplished by the application taking user input and combining it with static parameters to build a SQL query. The following examples are based on true stories, unfortunately.

Owing to the lack of input validation and connecting to the database on behalf of a superuser or the one who can create users, the attacker may create a superuser in your database.

Esempio 4-6. Splitting the result set into pages ... and making superusers (PostgreSQL and MySQL)

```
$offset = argv[0]; // beware, no input validation!
$query  = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";
// with PostgreSQL
$result = pg_exec($conn, $query);
// with MySQL
$result = mysql_query($query);
```

Normal users click on the ’next’, ’prev’ links where the `$offset` is encoded into the URL. The script expects that the incoming `$offset` is decimal number. However, someone tries to break in with appending urlencode()’d form of the following to the URL

```
// in case of PostgreSQL
0;
insert into pg_shadow(usename,usesysid,usesuper,usecatupd,passwd)
    select ’crack’, usesysid, ’t’,’t’,’crack’
    from pg_shadow where usename=’postgres’;
--

// in case of MySQL
0;
UPDATE user SET Password=PASSWORD(’crack’) WHERE user=’root’;
FLUSH PRIVILEGES;
```

If it happened, then the script would present a superuser access to him. Note that `0;` is to supply a valid offset to the original query and to terminate it.

> **Nota:** It is common technique to force the SQL parser to ignore the rest of the query written by the developer with `--` which is the comment sign in SQL.

A feasible way to gain passwords is to circumvent your search result pages. What the attacker needs only is to try if there is any submitted variable used in SQL statement which is not handled properly. These filters can be set commonly in a preceding form to customize `WHERE, ORDER BY, LIMIT` and `OFFSET` clauses in `SELECT` statements. If your database supports the `UNION` construct, the attacker may try to append an entire query to the original one to list passwords from an arbitrary table. Using encrypted password fields is strongly encouraged.

Esempio 4-7. Listing out articles ... and some passwords (any database server)

```
$query  = "SELECT id, name, inserted, size FROM products
                  WHERE size = ’$size’
                  ORDER BY $order LIMIT $limit, $offset;";
$result = odbc_exec($conn, $query);
```

The static part of the query can be combined with another `SELECT` statement which reveals all passwords:

```
’
union select ’1’, concat(uname||’-’||passwd) as name, ’1971-01-01’, ’0’ from usertable;
--
```

If this query (playing with the `'` and `--`) were assigned to one of the variables used in `$query`, the query beast awakened.

SQL UPDATEs are also subject to attacking your database. These queries are also threatened by chopping and appending an entirely new query to it. But the attacker might fiddle with the `SET` clause. In this case some schema information must be possessed to manipulate the query successfully. This can be acquired by examing the form variable names, or just simply brute forcing. There are not so many naming convention for fields storing passwords or usernames.

Esempio 4-8. From resetting a password ... to gaining more privileges (any database server)

```
$query = "UPDATE usertable SET pwd='$pwd' WHERE uid='$uid';";
```

But a malicious user sumbits the value `' or uid like'%admin%'; --` to `$uid` to change the admin's password, or simply sets `$pwd` to `"hehehe', admin='yes', trusted=100 "` (with a trailing space) to gain more privileges. Then, the query will be twisted:

```
// $uid == ' or uid like'%admin%'; --
$query = "UPDATE usertable SET pwd='...' WHERE uid='' or uid like '%admin%'; -
-";

// $pwd == "hehehe', admin='yes', trusted=100 "
$query = "UPDATE usertable SET pwd='hehehe', admin='yes', trusted=100 WHERE ...;"
```

A frightening example how operating system level commands can be accessed on some database hosts.

Esempio 4-9. Attacking the database host's operating system (MSSQL Server)

```
$query  = "SELECT * FROM products WHERE id LIKE '%$prod%'";
$result = mssql_query($query);
```

If attacker submits the value `a%' exec master..xp_cmdshell 'net user test testpass /ADD' --` to `$prod`, then the `$query` will be:

```
$query  = "SELECT * FROM products
                    WHERE id LIKE '%a%'
                    exec master..xp_cmdshell 'net user test testpass /ADD'-
-";
$result = mssql_query($query);
```

MSSQL Server executes the SQL statements in the batch including a command to add a new user to the local accounts database. If this application were running as sa and the MSSQLSERVER service is running with sufficient privileges, the attacker would now have an account with which to access this machine.

> **Nota:** Some of the examples above is tied to a specific database server. This does not mean that a similar attack is impossible against other products. Your database server may be so vulnerable in other manner.

Avoiding techniques

You may plead that the attacker must possess a piece of information about the database schema in most examples. You are right, but you never know when and how it can be taken out, and if it happens, your database may be exposed. If you are using an open source, or publicly available database handling package, which may belong to a content management system or forum, the intruders easily produce a copy of a piece of your code. It may be also a security risk if it is a poorly designed one.

These attacks are mainly based on exploiting the code not being written with security in mind. Never trust on any kind of input, especially which comes from the client side, even though it comes from a select box, a hidden input field or a cookie. The first example shows that such a blameless query can cause disasters.

- Never connect to the database as a superuser or as the database owner. Use always customized users with very limited privileges.
- Check if the given input has the expected data type. PHP has a wide range of input validating functions, from the simplest ones found in Variable Functions and in Character Type Functions (e.g. is_numeric(), ctype_digit() respectively) onwards the Perl compatible Regular Expressions support.
- If the application waits for numerical input, consider to verify data with is_numeric(), or silently change its type using settype(), or use its numeric representation by sprintf().

Esempio 4-10. A more secure way to compose a query for paging

```
settype($offset, ’integer’);
$query = "SELECT id, name FROM products ORDER BY name LIMIT 20 OFFSET $offset;";

// please note %d in the format string, using %s would be meaningless
$query = sprintf("SELECT id, name FROM products ORDER BY name LIMIT 20 OFF-
SET %d;",
                 $offset);
```

- Quote each non numeric user input which is passed to the database with addslashes() or addcslashes(). See the first example. As the examples shows, quotes burnt into the static part of the query is not enough, and can be easily hacked.

- Do not print out any database specific information, especially about the schema, by fair means or foul. See also Error Reporting and Error Handling and Logging Functions.
- You may use stored procedures and previously defined cursors to abstract data access so that users do not directly access tables or views, but this solution has another impacts.

Besides these, you benefit from logging queries either within your script or by the database itself, if it supports. Obviously, the logging is unable to prevent any harmful attempt, but it can be helpful to trace back which application has been circumvented. The log is not useful by itself, but through the information it contains. The more detail is generally better.

Error Reporting

With PHP security, there are two sides to error reporting. One is beneficial to increasing security, the other is detrimental.

A standard attack tactic involves profiling a system by feeding it improper data, and checking for the kinds, and contexts, of the errors which are returned. This allows the system cracker to probe for information about the server, to determine possible weaknesses. For example, if an attacker had gleaned information about a page based on a prior form submission, they may attempt to override variables, or modify them:

Esempio 4-11. Attacking Variables with a custom HTML page

```
<form method="post" action="attacktarget?username=badfoo&password=badfoo">
<input type="hidden" name="username" value="badfoo">
<input type="hidden" name="password" value="badfoo">
</form>
```

The PHP errors which are normally returned can be quite helpful to a developer who is trying to debug a script, indicating such things as the function or file that failed, the PHP file it failed in, and the line number which the failure occured in. This is all information that can be exploited. It is not uncommon for a php developer to use show_source(), highlight_string(), or highlight_file() as a debugging measure, but in a live site, this can expose hidden variables, unchecked syntax, and other dangerous information. Especially dangerous is running code from known sources with built-in debugging handlers, or using common debugging techniques. If the attacker can determine what general technique you are using, they may try to brute-force a page, by sending various common debugging strings:

Esempio 4-12. Exploiting common debugging variables

```
<form method="post" action="attacktarget?errors=Y&amp;showerrors=1"&debug=1">
<input type="hidden" name="errors" value="Y">
<input type="hidden" name="showerrors" value="1">
<input type="hidden" name="debug" value="1">
</form>
```

Regardless of the method of error handling, the ability to probe a system for errors leads to providing an attacker with more information.

For example, the very style of a generic PHP error indicates a system is running PHP. If the attacker was looking at an .html page, and wanted to probe for the back-end (to look for known weaknesses in the system), by feeding it the wrong data they may be able to determine that a system was built with PHP.

A function error can indicate whether a system may be running a specific database engine, or give clues as to how a web page or programmed or designed. This allows for deeper investigation into open database ports, or to look for specific bugs or weaknesses in a web page. By feeding different pieces of bad data, for example, an attacker can determine the order of authentication in a script, (from the line number errors) as well as probe for exploits that may be exploited in different locations in the script.

A filesystem or general PHP error can indicate what permissions the webserver has, as well as the structure and organization of files on the web server. Developer written error code can aggravate this problem, leading to easy exploitation of formerly "hidden" information.

There are three major solutions to this issue. The first is to scrutinize all functions, and attempt to compensate for the bulk of the errors. The second is to disable error reporting entirely on the running code. The third is to use PHP's custom error handling functions to create your own error handler. Depending on your security policy, you may find all three to be applicable to your situation.

One way of catching this issue ahead of time is to make use of PHP's own error_reporting(), to help you secure your code and find variable usage that may be dangerous. By testing your code, prior to deployment, with E_ALL, you can quickly find areas where your variables may be open to poisoning or modification in other ways. Once you are ready for deployment, by using E_NONE, you insulate your code from probing.

Esempio 4-13. Finding dangerous variables with E_ALL

```
<?php
if ($username) {  // Not initialized or checked before usage
    $good_login = 1;
}
if ($good_login == 1) { // If above test fails, not initialized or checked be-
fore usage
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Using Register Globals

One feature of PHP that can be used to enhance security is configuring PHP with register_globals = off. By turning off the ability for any user-submitted variable to be injected into PHP code, you can

reduce the amount of variable poisoning a potential attacker may inflict. They will have to take the additional time to forge submissions, and your internal variables are effectively isolated from user submitted data.

While it does slightly increase the amount of effort required to work with PHP, it has been argued that the benefits far outweigh the effort.

Esempio 4-14. Working without register_globals=off

```
<?php
if ($username) {  // can be forged by a user in get/post/cookies
    $good_login = 1;
}

if ($good_login == 1) { // can be forged by a user in get/post/cookies,
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

Esempio 4-15. Working with register_globals = off

```
<?php
if($HTTP_COOKIE_VARS[’username’]){
    // can only come from a cookie, forged or otherwise
    $good_login = 1;
    fpassthru ("/highly/sensitive/data/index.html");
}
?>
```

By using this wisely, it’s even possible to take preventative measures to warn when forging is being attempted. If you know ahead of time exactly where a variable should be coming from, you can check to see if submitted data is coming from an inappropriate kind of submission. While it doesn’t guarantee that data has not been forged, it does require an attacker to guess the right kind of forging.

Esempio 4-16. Detecting simple variable poisoning

```
<?php
if ($HTTP_COOKIE_VARS[’username’] &&
    !$HTTP_POST_VARS[’username’] &&
    !$HTTP_GET_VARS[’username’] ) {
    // Perform other checks to validate the user name...
    $good_login = 1;
    fpassthru ("/highly/sensitive/data/index.html");
} else {
   mail("admin@example.com", "Possible breakin attempt", $HTTP_SERVER_VARS[’REMOTE_ADDR’]
   echo "Security violation, admin has been alerted.";
   exit;
}
```

```
?>
```

Of course, simply turning off register_globals does not mean code is secure. For every piece of data that is submitted, it should also be checked in other ways.

User Submitted Data

The greatest weakness in many PHP programs is not inherent in the language itself, but merely an issue of code not being written with security in mind. For this reason, you should always take the time to consider the implications of a given piece of code, to ascertain the possible damage if an unexpected variable is submitted to it.

Esempio 4-17. Dangerous Variable Usage

```
<?php
// remove a file from the user’s home directory... or maybe
// somebody else’s?
unlink ($evil_var);

// Write logging of their access... or maybe an /etc/passwd entry?
fputs ($fp, $evil_var);

// Execute something trivial.. or rm -rf *?
system ($evil_var);
exec ($evil_var);

?>
```

You should always carefully examine your code to make sure that any variables being submitted from a web browser are being properly checked, and ask yourself the following questions:

- Will this script only affect the intended files?
- Can unusual or undesirable data be acted upon?
- Can this script be used in unintended ways?
- Can this be used in conjunction with other scripts in a negative manner?
- Will any transactions be adequately logged?

By adequately asking these questions while writing the script, rather than later, you prevent an unfortunate re-write when you need to increase your security. By starting out with this mindset, you won’t guarantee the security of your system, but you can help improve it.

You may also want to consider turning off register_globals, magic_quotes, or other convenience settings which may confuse you as to the validity, source, or value of a given variable. Working with PHP in error_reporting(E_ALL) mode can also help warn you about variables being used before they are checked or initialized (so you can prevent unusual data from being operated upon).

Hiding PHP

In general, security by obscurity is one of the weakest forms of security. But in some cases, every little bit of extra security is desirable.

A few simple techniques can help to hide PHP, possibly slowing down an attacker who is attempting to discover weaknesses in your system. By setting expose_php = off in your `php.ini` file, you reduce the amount of information available to them.

Another tactic is to configure web servers such as apache to parse different filetypes through PHP, either with an .htaccess directive, or in the apache configuration file itself. You can then use misleading file extensions:

Esempio 4-18. Hiding PHP as another language

```
# Make PHP code look like other code types
AddType application/x-httpd-php .asp .py .pl
```

Or obscure it completely:

Esempio 4-19. Using unknown types for PHP extensions

```
# Make PHP code look like unknown types
AddType application/x-httpd-php .bop .foo .133t
```

Or hide it as html code, which has a slight performance hit because all html will be parsed through the PHP engine:

Esempio 4-20. Using html types for PHP extensions

```
# Make all PHP code look like html
AddType application/x-httpd-php .htm .html
```

For this to work effectively, you must rename your PHP files with the above extensions. While it is a form of security through obscurity, it's a minor preventative measure with few drawbacks.

Keeping Current

PHP, like any other large system, is under constant scrutiny and improvement. Each new version will often include both major and minor changes to enhance and repair security flaws, configuration mishaps, and other issues that will affect the overall security and stability of your system.

Like other system-level scripting languages and programs, the best approach is to update often, and maintain awareness of the latest versions and their changes.

Parte II. Struttura del Linguaggio

Capitolo 5. Sintassi Fondamentale

Modi per uscire dalla modalità HTML

Quando il PHP inizia a manipolare un file, produrrà in uscita solamente il testo che trova. Così se si ha un file HTML, e si modifica l'estensione in .php, il file continuerà ad essere visibile.

Se si vogliono inserire delle istruzioni PHP in un certo punto del file, occorre indicarlo al php, entrando nella "modalità PHP" in uno dei seguenti modi:

Esempio 5-1. Metodi per uscire dalla modalità HTML

```
1.  <? echo ("questo è il più semplice, ovvero come istruzione SGML\n"); ?>
    <?= espressione ?>  Questa è un'abbreviazione per "<? echo espressione ?>"

2.  <?php echo("se si vogliono produrre documenti XHTML o XML, si utilizzi questo modo\n"

3.  <script language="php">
        echo ("alcuni editor (tipo FrontPage) non
               amano le istruzioni di elaborazione");
    </script>

4.  <% echo ("Opzionalmente puoi utilizzare tag nello stile ASP"); %>
    <%= $variable; # Questo è una abbreviazione per "<%echo .." %>
```

Il primo è disponibile solo se sono stati abilitati i tags abbreviati. Ciò può essere impostato abilitando nel file di configurazione del PHP l'opzione short_open_tag, oppure compilando il PHP utilizzando l'opzione --enable-short-tags nel comando **configure**.

Il secondo modo è il metodo generalmente preferito, in quanto consente alla prossima generazione di XHTML di essere facilmente implementato con il PHP.

Il quarto modo è disponibile solo se sono stati attivati nel file di configurazione i tag in stile ASP tramite l'opzione asp_tags.

> **Nota:** Il supporto per i tag nello stile ASP è stato aggiunto nella versione 3.0.4.

I tag di chiusura del blocco includono, se presente, il carattere di newline immediatamente sucessivo.

PHP allows you to use structures like this:

Esempio 5-2. Advanced escaping

```
<?php

if (boolean-expression) {
    ?>
<strong>This is true.</strong>
    <?php
} else {
    ?>
<strong>This is false.</strong>
    <?php
}
```

```
    ?>
```

This works as expected, because PHP handles text within ?> and <?php as an echo() statement.

Separazione delle istruzioni

Le istruzioni sono separate come nel C o in perl - ogni istruzione termina con un punto e virgola.

Il tag di chiusura (?>) implica anche la fine di un’istruzione, perciò le seguenti sono equivalenti:

```
<?php
    echo "Questo ‘ un test";
?>

<?php echo "Questo ‘ un test" ?>
```

Commenti

PHP supporta i commenti dei linguaggi ’C’, ’C++’ e della shell Unix. Per esempio:

```
<?php
    echo "Questo ‘ un test"; // Questo è un commento su una linea nella stile c++
    /* Questo è un commento su più linee
       ancora un’altra linea di commento */
    echo "Questo è un altro test";
    echo "Un ultimo test"; # Questo è un commento stile shell Unix
?>
```

Lo stile di commento su "una linea", attualmente commenta solo fino alla fine della linea o del blocco corrente di codice PHP.

```
<h1>Questo è un <?# echo "semplice";?> esempio.</h1>
<p>L’intestazione qui sopra dirà ’Questo è un esempio’.
```

Occorre fare attenzione nel non annidare i commenti di stile C, situazione che si presenta quando si commentano larghi blocchi di codice.

```
<?php
 /*
    echo "Questo è un test"; /* Questo commento causerà dei problemi */
 */
?>
```

Capitolo 6. Types

Introduction

PHP supports eight primitive types.

Four scalar types:

- boolean
- integer
- floating-point number (float)
- string

Two compound types:

- array
- object

And finally two special types:

- resource
- NULL

> **Nota:** In this manual you'll often find `mixed` parameters. This pseudo-type indicates multiple possibilities for that parameter.

The type of a variable is usually not set by the programmer; rather, it is decided at runtime by PHP depending on the context in which that variable is used.

> **Nota:** If you want to check out the type and value of a certain expression, use var_dump().
>
> If you simply want a human-readable representation of the type for debugging, use gettype(). To check for a certain type, do *not* use gettype(), but use the `is_type` functions.

If you would like to force a variable to be converted to a certain type, you may either cast the variable or use the settype() function on it.

Note that a variable may behave in different manners in certain situations, depending on what type it is at the time. For more information, see the section on Type Juggling.

Booleans

This is the easiest type. A boolean expresses a truth value. It can be either `TRUE` or `FALSE`.

> **Nota:** The boolean type was introduced in PHP 4.

Syntax

To specify a boolean literal, use either the keyword `TRUE` or `FALSE`. Both are case-insensitive.

```
$foo = True; // assign the value TRUE to $foo
```

Usually you use some kind of operator which returns a boolean value, and then pass it on to a control structure.

```
// == is an operator which returns a boolean
if ($action == "show_version") {
    echo "The version is 1.23";
}

// this is not necessary:
if ($show_separators == TRUE) {
    echo "<hr>\n";
}

// because you can simply type this:
if ($show_separators) {
    echo "<hr>\n";
}
```

Converting to boolean

To explicitly convert a value to boolean, use either the `(bool)` or the `(boolean)` cast. However, in most cases you do not need to use the cast, since a value will be automatically converted if an operator, function or control structure requires a boolean argument.

See also Type Juggling.

When converting to boolean, the following values are considered `FALSE`:

- the boolean `FALSE`
- the integer 0 (zero)
- the float 0.0 (zero)
- the empty string, and the string "0"
- an array with zero elements
- an object with zero elements
- the special type NULL (including unset variables)

Every other value is considered `TRUE` (including any resource).

> **Attenzione**
>
> `-1` is considered `TRUE`, like any other non-zero (whether negative or positive) number!

Integers

An integer is a number of the set Z = {..., -2, -1, 0, 1, 2, ...}.

See also: Arbitrary length integers and Floating point numbers

Syntax

Integers can be specified in decimal (10-based), hexadecimal (16-based) or octal (8-based) notation, optionally preceded by a sign (- or +).

If you use the octal notation, you must precede the number with a `0` (zero), to use hexadecimal notation precede the number with `0x`.

Esempio 6-1. Integer literals

```
$a = 1234; # decimal number
$a = -123; # a negative number
$a = 0123; # octal number (equivalent to 83 decimal)
$a = 0x1A; # hexadecimal number (equivalent to 26 decimal)
```

The size of an integer is platform-dependent, although a maximum value of about two billion is the usual value (that's 32 bits signed). PHP does not support unsigned integers.

Integer overflow

If you specify a number beyond the bounds of the integer type, it will be interpreted as a float instead. Also, if you perform an operation that results in a number beyond the bounds of the integer type, a float will be returned instead.

```
$large_number =  2147483647;
var_dump($large_number);
// output: int(2147483647)

$large_number =  2147483648;
var_dump($large_number);
// output: float(2147483648)

// this goes also for hexadecimal specified integers:
var_dump( 0x80000000 );
```

```
// output: float(2147483648)

$million = 1000000;
$large_number =  50000 * $million;
var_dump($large_number);
// output: float(50000000000)
```

> **Attenzione**
>
> Unfortunately, there was a bug in PHP so that this does not always work correctly when there are negative numbers involved. For example: when you do `-50000 * $million`, the result will be `-429496728`. However, when both operands are positive there is no problem.
>
> This is solved in PHP 4.1.0.

There is no integer division operator in PHP. `1/2` yields the float `0.5`.

```
var_dump( 25/7 );
// output: float(3.5714285714286)
```

Converting to integer

To explicitly convert a value to integer, use either the `(int)` or the `(integer)` cast. However, in most cases you do not need to use the cast, since a value will be automatically converted if an operator, function or control structure requires a integer argument.

See also type-juggling.

From booleans

`FALSE` will yield `0` (zero), and `TRUE` will yield `1` (one).

From floating point numbers

When converting from float to integer, the number will be rounded *towards zero*.

If the float is beyond the boundaries of integer (usually `+/- 2.15e+9 = 2^31`), the result is undefined, since the float hasn't got enough precision to give an exact integer result. No warning, not even a notice will be issued in this case!

Attenzione

Never cast an unknown fraction to integer, as this can sometimes lead to unexpected results.

```
echo (int) ( (0.1+0.7) * 10 ); // echoes 7!
```

See for more information the warning about float-precision.

From strings

See String conversion

From other types

Cautela

Behaviour of converting to integer is undefined for other types. Currently, the behaviour is the same as if the value was first converted to boolean. However, do *not* rely on this behaviour, as it can change without notice.

Floating point numbers

Floating point numbers (AKA "floats", "doubles" or "real numbers") can be specified using any of the following syntaxes:

```
$a = 1.234; $a = 1.2e3; $a = 7E-10;
```

The size of a float is platform-dependent, although a maximum of ~1.8e308 with a precision of roughly 14 decimal digits is a common value (that’s 64 bit IEEE format).

> **Floating point precision**
>
> It is quite usual that simple decimal fractions like `0.1` or `0.7` cannot be converted into their internal binary counterparts without a little loss of precision. This can lead to confusing results: for example, `floor((0.1+0.7)*10)` will usually return `7` instead of the expected `8` as the result of the internal representation really being something like `7.9999999999....`
>
> This is related to the fact that it is impossible to exactly express some fractions in decimal notation with a finite number of digits. For instance, `1/3` in decimal form becomes `0.3333333. . ..`
>
> So never trust floating number results to the last digit and never compare floating point numbers for equality. If you really need higher precision, you should use the arbitrary precision math functions or gmp functions instead.

Strings

A string is series of characters. In PHP, a character is the same as a byte, that is, there are exactly 256 different characters possible. This also implies that PHP has no native support of Unicode.

> **Nota:** It is no problem for a string to become very large. There is no practical bound to the size of strings imposed by PHP, so there is no reason at all to worry about long strings.

Syntax

A string literal can be specified in three different ways.

- single quoted
- double quoted
- heredoc syntax

Single quoted

The easiest way to specify a simple string is to enclose it in single quotes (the character `'`).

To specify a literal single quote, you will need to escape it with a backslash (\), like in many other languages. If a backslash needs to occur before a single quote or at the end of the string, you need to double it. Note that if you try to escape any other character, the backslash too will be printed! So usually there is no need to escape the backslash itself.

> **Nota:** In PHP 3, a warning will be issued at the `E_NOTICE` level when this happens.

> **Nota:** Unlike the two other syntaxes, variables will *not* be expanded when they occur in single quoted strings.

```
echo 'this is a simple string';
echo 'You can also have embedded newlines in strings,
like this way.';
echo 'Arnold once said: "I\'ll be back"';
// output: ... "I'll be back"
echo 'Are you sure you want to delete C:\\*.*?';
// output: ... delete C:\*.*?
echo 'Are you sure you want to delete C:\*.*?';
// output: ... delete C:\*.*?
echo 'I am trying to include at this point: \n a newline';
// output: ... this point: \n a newline
```

Double quoted

If the string is enclosed in double-quotes ("), PHP understands more escape sequences for special characters:

Tabella 6-1. Escaped characters

sequence	meaning
`\n`	linefeed (LF or 0x0A (10) in ASCII)
`\r`	carriage return (CR or 0x0D (13) in ASCII)
`\t`	horizontal tab (HT or 0x09 (9) in ASCII)
`\\`	backslash
`\$`	dollar sign
`\"`	double-quote
`\[0-7]{1,3}`	the sequence of characters matching the regular expression is a character in octal notation
`\x[0-9A-Fa-f]{1,2}`	the sequence of characters matching the regular expression is a character in hexadecimal notation

Again, if you try to escape any other character, the backslash will be printed too!

But the most important pre of double-quoted strings is the fact that variable names will be expanded. See string parsing for details.

Heredoc

Another way to delimit strings is by using here doc syntax ("<<<"). One should provide an identifier after <<<, then the string, and then the same identifier to close the quotation.

The closing identifier *must* begin in the first column of the line. Also, the identifier used must follow the same naming rules as any other label in PHP: it must contain only alphanumeric characters and underscores, and must start with a non-digit character or underscore.

Attenzione

It is very important to note that the line with the closing identifier contains no other characters, except *possibly* a semicolon (`;`). That means especially that the identifier *may not be indented*, and there may not be any spaces or tabs after or before the semicolon.

Probably the nastiest gotcha is that there may also not be a carriage return (`\r`) at the end of the line, only a form feed, AKA newline (`\n`). Since Microsoft Windows uses the sequence `\r\n` as a line terminator, your heredoc may not work if you write your script in a Windows editor. However, most programming editors provide a way to save your files with a UNIX line terminator.

Here doc text behaves just like a double-quoted string, without the double-quotes. This means that you do not need to escape quotes in your here docs, but you can still use the escape codes listed above. Variables are expanded, but the same care must be taken when expressing complex variables inside a here doc as with strings.

Esempio 6-2. Here doc string quoting example

```
<?php
$str = <<<EOD
Example of string
spanning multiple lines
using heredoc syntax.
EOD;

/* More complex example, with variables. */
class foo
{
    var $foo;
    var $bar;

    function foo()
    {
        $this->foo = ’Foo’;
        $this->bar = array(’Bar1’, ’Bar2’, ’Bar3’);
    }
}

$foo = new foo();
$name = ’MyName’;

echo <<<EOT
My name is "$name". I am printing some $foo->foo.
Now, I am printing some {$foo->bar[1]}.
This should print a capital ’A’: \x41
EOT;
?>
```

Nota: Here doc support was added in PHP 4.

Variable parsing

When a string is specified in double quotes or with heredoc, variables are parsed within it.

There are two types of syntax, a simple one and a complex one. The simple syntax is the most common and convenient, it provides a way to parse a variable, an array value, or an object property.

The complex syntax was introduced in PHP 4, and can by recognised by the curly braces surrounding the expression.

Simple syntax

If a dollar sign (`$`) is encountered, the parser will greedily take as much tokens as possible to form a valid variable name. Enclose the variable name in curly braces if you want to explicitly specify the end of the name.

```
$beer = 'Heineken';
echo "$beer's taste is great"; // works, "'" is an invalid character for varnames
echo "He drunk some $beers"; // won't work, 's' is a valid character for varnames
echo "He drunk some ${beer}s"; // works
```

Similarly, you can also have an array index or an object property parsed. With array indices, the closing square bracket (`]`) marks the end of the index. For object properties the same rules apply as to simple variables, though with object properties there doesn't exist a trick like the one with variables.

```
$fruits = array( 'strawberry' => 'red' , 'banana' => 'yellow' );

// note that this works differently outside string-quotes.
echo "A banana is $fruits[banana].";

echo "This square is $square->width meters broad.";

// Won't work. For a solution, see the complex syntax.
echo "This square is $square->width00 centimeters broad.";
```

For anything more complex, you should use the complex syntax.

Complex (curly) syntax

This isn't called complex because the syntax is complex, but because you can include complex expressions this way.

In fact, you can include any value that is in the namespace in strings with this syntax. You simply write the expression the same way as you would outside the string, and then include it in `{` and `}`.

Since you can't escape '{', this syntax will only be recognised when the $ is immediately following the {. (Use "{\$" or "\{$" to get a literal "{$"). Some examples to make it clear:

```
$great = 'fantastic';
echo "This is { $great}"; // won't work, outputs: This is { fantastic}
echo "This is {$great}";  // works, outputs: This is fantastic
echo "This square is {$square->width}00 centimeters broad.";
echo "This works: {$arr[4][3]}";

// This is wrong for the same reason
// as $foo[bar] is wrong outside a string.
echo "This is wrong: {$arr[foo][3]}";

echo "You should do it this way: {$arr['foo'][3]}";
echo "You can even write {$obj->values[3]->name}";
echo "This is the value of the var named $name: {${$name}}";
```

String access by character

Characters within strings may be accessed by specifying the zero-based offset of the desired character after the string in curly braces.

> **Nota:** For backwards compatibility, you can still use the array-braces. However, this syntax is deprecated as of PHP 4.

Esempio 6-3. Some string examples

```
<?php
/* Assigning a string. */
$str = "This is a string";

/* Appending to it. */
$str = $str . " with some more text";

/* Another way to append, includes an escaped newline. */
$str .= " and a newline at the end.\n";

/* This string will end up being '<p>Number: 9</p>' */
$num = 9;
$str = "<p>Number: $num</p>";

/* This one will be '<p>Number: $num</p>' */
$num = 9;
$str = '<p>Number: $num</p>';
```

```
/* Get the first character of a string  */
$str = 'This is a test.';
$first = $str{0};

/* Get the last character of a string. */
$str = 'This is still a test.';
$last = $str{strlen($str)-1};
?>
```

Useful functions

Strings may be concatenated using the '.' (dot) operator. Note that the '+' (addition) operator will not work for this. Please see String operators for more information.

There are a lot of useful functions for string modification.

See the string functions section for general functions, the regular expression functions for advanced find&replacing (in two tastes: Perl and POSIX extended).

There are also functions for URL-strings, and functions to encrypt/decrypt strings (mcrypt and mhash).

Finally, if you still didn't find what you're looking for, see also the character type functions.

String conversion

When a string is evaluated as a numeric value, the resulting value and type are determined as follows.

The string will evaluate as a float if it contains any of the characters '.', 'e', or 'E'. Otherwise, it will evaluate as an integer.

The value is given by the initial portion of the string. If the string starts with valid numeric data, this will be the value used. Otherwise, the value will be 0 (zero). Valid numeric data is an optional sign, followed by one or more digits (optionally containing a decimal point), followed by an optional exponent. The exponent is an 'e' or 'E' followed by one or more digits.

When the first expression is a string, the type of the variable will depend on the second expression.

```
$foo = 1 + "10.5";              // $foo is float (11.5)
$foo = 1 + "-1.3e3";            // $foo is float (-1299)
$foo = 1 + "bob-1.3e3";         // $foo is integer (1)
$foo = 1 + "bob3";              // $foo is integer (1)
$foo = 1 + "10 Small Pigs";     // $foo is integer (11)
$foo = 1 + "10 Little Piggies"; // $foo is integer (11)
$foo = "10.0 pigs " + 1;        // $foo is integer (11)
$foo = "10.0 pigs " + 1.0;      // $foo is float (11)
```

For more information on this conversion, see the Unix manual page for strtod(3).

If you would like to test any of the examples in this section, you can cut and paste the examples and insert the following line to see for yourself what's going on:

```
echo "\$foo==$foo; type is " . gettype ($foo) . "<br>\n";
```

Arrays

An array in PHP is actually an ordered map. A map is a type that maps *values* to *keys*. This type is optimized in several ways, so you can use it as a real array, or a list (vector), hashtable (which is an implementation of a map), dictionary, collection, stack, queue and probably more. Because you can have another PHP-array as a value, you can also quite easily simulate trees.

Explanation of those structures is beyond the scope of this manual, but you'll find at least one example for each of those structures. For more information about those structures, we refer you to external literature about this broad topic.

Syntax

Specifying with array()

An array can be created by the array() language-construct. It takes a certain number of comma-separated `key => value` pairs.

A `key` is either a nonnegative integer or a string. If a key is the standard representation of a non-negative integer, it will be interpreted as such (i.e. `'8'` will be interpreted as `8`, while `'08'` will be interpreted as `'08'`).

A value can be anything.

If you omit a key, the maximum of the integer-indices is taken, and the new key will be that maximum + 1. If no integer-indices exist yet, the key will be `0` (zero). If you specify a key that already has a value assigned to it, that value will be overwritten.

```
array( [key =>] value
     , ...
     )
// key is either string or nonnegative integer
// value can be anything
```

Creating/modifying with square-bracket syntax

You can also modify an existing array, by explicitly setting values.

This is done by assigning values to the array while specifying the key in brackets. You can also omit the key, add an empty pair of brackets ("`[]`") to the variable-name in that case.

```
$arr[key] = value;
$arr[] = value;
// key is either string or nonnegative integer
// value can be anything
```

If `$arr` doesn't exist yet, it will be created. So this is also an alternative way to specify an array. To change a certain value, just assign a new value to it. If you want to remove a key/value pair, you need to unset() it.

Useful functions

There are quite some useful function for working with arrays, see the array-functions section.

> **Nota:** The unset() function allows unsetting keys of an array. Be aware that the array will NOT be reindexed.

```
$a = array( 1 => 'one', 2 => 'two', 3 => 'three' );
unset( $a[2] );
/* will produce an array that would have been defined as
   $a = array( 1=>'one', 3=>'three');
   and NOT
   $a = array( 1 => 'one', 2 => 'three');
*/
```

The foreach control structure exists specificly for arrays. It provides an easy way to traverse an array.

Array do's and don'ts

Why is `$foo[bar]` wrong?

You should always use quotes around an associative array index. For example, use $foo['bar'] and not $foo[bar]. But why is $foo[bar] wrong? You might have seen the following syntax in old scripts:

```
$foo[bar] = 'enemy';
echo $foo[bar];
// etc
```

This is wrong, but it works. Then, why is it wrong? The reason is that this code has an undefined constant (bar) rather than a string ('bar' - notice the quotes), and PHP may in future define constants

which, unfortunately for your code, have the same name. It works, because the undefined constant gets converted to a string of the same name.

As stated in the syntax section, there must be an expression between the square brackets ('`[`' and '`]`'). That means that you can write things like this:

```
echo $arr[ foo(true) ];
```

This is an example of using a function return value as the array index. PHP knows also about constants, and you may have seen the `E_*` before.

```
$error_descriptions[E_ERROR] = "A fatal error has occured";
$error_descriptions[E_WARNING] = "PHP issued a warning";
$error_descriptions[E_NOTICE] = "This is just an informal notice";
```

Note that `E_ERROR` is also a valid identifier, just like `bar` in the first example. But the last example is in fact the same as writing:

```
$error_descriptions[1] = "A fatal error has occured";
$error_descriptions[2] = "PHP issued a warning";
$error_descriptions[8] = "This is just an informal notice";
```

because `E_ERROR` equals `1`, etc.

Then, how is it possible that `$foo[bar]` works? It works, because `bar` is due to its syntax expected to be a constant expression. However, in this case no constant with the name `bar` exists. PHP now assumes that you meant `bar` literally, as the string `"bar"`, but that you forgot to write the quotes.

So why is it bad then?

At some point in the future, the PHP team might want to add another constant or keyword, and then you get in trouble. For example, you already cannot use the words `empty` and `default` this way, since they are special reserved keywords.

> **Nota:** When you turn error_reporting to `E_ALL`, you will see that PHP generates notices whenever an `index` is used which is not defined (put the line `error_reporting(E_ALL);` in your script).

> **Nota:** Inside a double-quoted string, an other syntax is valid. See variable parsing in strings for more details.

Examples

The array type in PHP is very versatile, so here will be some examples to show you the full power of arrays.

```
// this
$a = array( 'color' => 'red'
          , 'taste' => 'sweet'
          , 'shape' => 'round'
          , 'name'  => 'apple'
          ,            4        // key will be 0
          );

// is completely equivalent with
$a['color'] = 'red';
$a['taste'] = 'sweet';
$a['shape'] = 'round';
$a['name'] = 'apple';
$a[]        = 4;        // key will be 0

$b[] = 'a';
$b[] = 'b';
$b[] = 'c';
// will result in the array array( 0 => 'a' , 1 => 'b' , 2 => 'c' ),
// or simply array('a', 'b', 'c')
```

Esempio 6-4. Using array()

```
// Array as (property-)map
$map = array( 'version'    => 4
            , 'OS'         => 'Linux'
            , 'lang'       => 'english'
            , 'short_tags' => true
            );

// strictly numerical keys
$array = array( 7
              , 8
              , 0
              , 156
              , -10
              );
// this is the same as array( 0 => 7, 1 => 8, ...)

$switching = array(         10 // key = 0
                  , 5    =>  6
                  , 3    =>  7
                  , 'a'  =>  4
                  ,         11 // key = 6 (maximum of integer-indices was 5)
                  , '8'  =>  2 // key = 8 (integer!)
```

```
                  , ’02’ => 77 // key = ’02’
                  , 0    => 12 // the value 10 will be overwritten by 12
                  );

// empty array
$empty = array();
```

Esempio 6-5. Collection

```
$colors = array(’red’,’blue’,’green’,’yellow’);

foreach ( $colors as $color ) {
    echo "Do you like $color?\n";
}

/* output:
Do you like red?
Do you like blue?
Do you like green?
Do you like yellow?
*/
```

Note that it is currently not possible to change the values of the array directly in such a loop. A workaround is the following:

Esempio 6-6. Collection

```
foreach ($colors as $key => $color) {
    // won’t work:
    //$color = strtoupper($color);

    //works:
    $colors[$key] = strtoupper($color);
}
print_r($colors);

/* output:
Array
(
    [0] => RED
    [1] => BLUE
    [2] => GREEN
    [3] => YELLOW
)
*/
```

This example creates a one-based array.

Esempio 6-7. One-based index

```
$firstquarter  = array(1 => 'January', 'February', 'March');
print_r($firstquarter);

/* output:
Array
(
    [1] => 'January'
    [2] => 'February'
    [3] => 'March'
)
*/
```

Esempio 6-8. Filling real array

```
// fill an array with all items from a directory
$handle = opendir('.');
while ($file = readdir($handle))
{
    $files[] = $file;
}
closedir($handle);
```

Arrays are ordered. You can also change the order using various sorting-functions. See array-functions for more information.

Esempio 6-9. Sorting array

```
sort($files);
print_r($files);
```

Because the value of an array can be everything, it can also be another array. This way you can make recursive and multi-dimensional arrays.

Esempio 6-10. Recursive and multi-dimensional arrays

```
$fruits = array ( "fruits"  => array ( "a" => "orange"
                                     , "b" => "banana"
                                     , "c" => "apple"
                                     )
                , "numbers" => array ( 1
```

```
                                            , 2
                                            , 3
                                            , 4
                                            , 5
                                            , 6
                                            )
                    , "holes"   => array (      "first"
                                            , 5 => "second"
                                            ,      "third"
                                            )
                    );
```

Objects

Object Initialization

To initialize an object, you use the `new` statement to instantiate the object to a variable.

```
<?php
class foo
{
    function do_foo()
    {
        echo "Doing foo.";
    }
}

$bar = new foo;
$bar->do_foo();
?>
```

For a full discussion, please read the section Classes and Objects.

Resource

A resource is a special variable, holding a reference to an external resource. Resources are created and used by special functions. See the appendix for a listing of all these functions and the corresponding resource types.

> **Nota:** The resource type was introduced in PHP 4

Freeing resources

Due to the reference-counting system introduced with PHP4's Zend-engine, it is automatically detected when a resource is no longer referred to (just like Java). When this is the case, all resources that were in use for this resource are made free by the garbage collector. For this reason, it is rarely ever necessary to free the memory manually by using some free_result function.

> **Nota:** Persistent database-links are special, they are *not* destroyed by the gc. See also persistent links

NULL

The special `NULL` value represents that a variable has no value. `NULL` is the only possible value of type NULL.

> **Nota:** The null type was introduced in PHP 4

Syntax

There is only one value of type `NULL`, and that is the case-insensitive keyword `NULL`.

```
$var = NULL;
```

Type Juggling

PHP does not require (or support) explicit type definition in variable declaration; a variable's type is determined by the context in which that variable is used. That is to say, if you assign a string value to variable *var*, *var* becomes a string. If you then assign an integer value to *var*, it becomes an integer.

An example of PHP's automatic type conversion is the addition operator '+'. If any of the operands is a float, then all operands are evaluated as floats, and the result will be a float. Otherwise, the operands will be interpreted as integers, and the result will also be an integer. Note that this does NOT change the types of the operands themselves; the only change is in how the operands are evaluated.

```
$foo = "0";  // $foo is string (ASCII 48)

$foo += 2;   // $foo is now an integer (2)
$foo = $foo + 1.3;  // $foo is now a float (3.3)
$foo = 5 + "10 Little Piggies"; // $foo is integer (15)
$foo = 5 + "10 Small Pigs";     // $foo is integer (15)
```

If the last two examples above seem odd, see String conversion.

If you wish to force a variable to be evaluated as a certain type, see the section on Type casting. If you wish to change the type of a variable, see settype().

If you would like to test any of the examples in this section, you can use the var_dump() function.

Nota: The behaviour of an automatic conversion to array is currently undefined.

```
$a = 1;       // $a is an integer
$a[0] = "f";  // $a becomes an array, with $a[0] holding "f"
```

While the above example may seem like it should clearly result in $a becoming an array, the first element of which is 'f', consider this:

```
$a = "1";     // $a is a string
$a[0] = "f";  // What about string offsets? What happens?
```

Since PHP supports indexing into strings via offsets using the same syntax as array indexing, the example above leads to a problem: should $a become an array with its first element being "f", or should "f" become the first character of the string $a?

For this reason, as of PHP 3.0.12 and PHP 4.0b3-RC4, the result of this automatic conversion is considered to be undefined. Fixes are, however, being discussed.

Type Casting

Type casting in PHP works much as it does in C: the name of the desired type is written in parentheses before the variable which is to be cast.

```
$foo = 10;   // $foo is an integer
$bar = (float) $foo;   // $bar is a float
```

The casts allowed are:

- (int), (integer) - cast to integer
- (bool), (boolean) - cast to boolean
- (float), (double), (real) - cast to float
- (string) - cast to string
- (array) - cast to array

- (object) - cast to object

> **Nota:** Instead of casting a variable to string, you can also enclose the variable in double quotes.

Note that tabs and spaces are allowed inside the parentheses, so the following are functionally equivalent:

```
$foo = (int) $bar;
$foo = ( int ) $bar;
```

It may not be obvious exactly what will happen when casting between certain types. For more info, see these sections:

- Converting to boolean
- Converting to integer

When casting or forcing a conversion from array to string, the result will be the word `Array`. When casting or forcing a conversion from object to string, the result will be the word `Object`.

When casting from a scalar or a string variable to an array, the variable will become the first element of the array:

```
$var = 'ciao';
$arr = (array) $var;
echo $arr[0];  // outputs 'ciao'
```

When casting from a scalar or a string variable to an object, the variable will become an attribute of the object; the attribute name will be 'scalar':

```
$var = 'ciao';
$obj = (object) $var;
echo $obj->scalar;  // outputs 'ciao'
```

Capitolo 7. Variables

Basics

Variables in PHP are represented by a dollar sign followed by the name of the variable. The variable name is case-sensitive.

Variable names follow the same rules as other labels in PHP. A valid variable name starts with a letter or underscore, followed by any number of letters, numbers, or underscores. As a regular expression, it would be expressed thus: '[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*'

> **Nota:** For our purposes here, a letter is a-z, A-Z, and the ASCII characters from 127 through 255 (0x7f-0xff).

```
$var = "Bob";
$Var = "Joe";
echo "$var, $Var";      // outputs "Bob, Joe"

$4site = 'not yet';     // invalid; starts with a number
$_4site = 'not yet';    // valid; starts with an underscore
$täyte = 'mansikka';    // valid; 'ä' is ASCII 228.
```

In PHP 3, variables are always assigned by value. That is to say, when you assign an expression to a variable, the entire value of the original expression is copied into the destination variable. This means, for instance, that after assigning one variable's value to another, changing one of those variables will have no effect on the other. For more information on this kind of assignment, see Expressions.

PHP 4 offers another way to assign values to variables: *assign by reference*. This means that the new variable simply references (in other words, "becomes an alias for" or "points to") the original variable. Changes to the new variable affect the original, and vice versa. This also means that no copying is performed; thus, the assignment happens more quickly. However, any speedup will likely be noticed only in tight loops or when assigning large arrays or objects.

To assign by reference, simply prepend an ampersand (&) to the beginning of the variable which is being assigned (the source variable). For instance, the following code snippet outputs 'My name is Bob' twice:

```
<?php
$foo = 'Bob';              // Assign the value 'Bob' to $foo
$bar = &$foo;              // Reference $foo via $bar.
$bar = "My name is $bar";  // Alter $bar...
echo $bar;
echo $foo;                 // $foo is altered too.
?>
```

One important thing to note is that only named variables may be assigned by reference.

```
<?php
$foo = 25;
$bar = &$foo;      // This is a valid assignment.
$bar = &(24 * 7);  // Invalid; references an unnamed expression.

function test()
{
   return 25;
}

$bar = &test();    // Invalid.
?>
```

Predefined variables

PHP provides a large number of predefined variables to any script which it runs. Many of these variables, however, cannot be fully documented as they are dependent upon which server is running, the version and setup of the server, and other factors. Some of these variables will not be available when PHP is run on the command line. For a listing of these variables, please see the section Predefined variables.

> **Attenzione**
>
> In PHP 4.2.0 and later, the default set of predefined variables which are available in the global scope has changed. Individual input and server variables are *by default* no longer placed directly into the global scope; rather, they are placed into the following superglobal arrays.
>
> You can still force the old behaviour by setting register_globals to 'On' in your `php.ini` file.
>
> For more information and background on this change, please see the PHP 4.1.0 Release Announcement (http://www.php.net/release_4_1_0.php).

From version 4.1.0 onward, PHP provides a set of predefined arrays containing variables from the web server (if applicable), the environment, and user input. These new arrays are rather special in that they are automatically global--i.e., automatically available in every scope. For this reason, they are often known as 'autoglobals' or 'superglobals'. (There is no mechanism in PHP for user-defined superglobals.) The superglobals are listed below; however, for a listing of their contents and further discussion on PHP predefined variables and their natures, please see the section Predefined variables.

PHP Superglobals

$GLOBALS

Contains a reference to every variable which is currently available within the global scope of the script. The keys of this array are the names of the global variables.

$_SERVER

Variables set by the web server or otherwise directly related to the execution environment of the current script. Analogous to the old `$HTTP_SERVER_VARS` array (which is still available, but deprecated).

$_GET

Variables provided to the script via HTTP GET. Analogous to the old `$HTTP_GET_VARS` array (which is still available, but deprecated).

$_POST

Variables provided to the script via HTTP POST. Analogous to the old `$HTTP_POST_VARS` array (which is still available, but deprecated).

$_COOKIE

Variables provided to the script via HTTP cookies. Analogous to the old `$HTTP_COOKIE_VARS` array (which is still available, but deprecated).

$_FILES

Variables provided to the script via HTTP post file uploads. Analogous to the old `$HTTP_POST_FILES` array (which is still available, but deprecated). See POST method uploads for more information.

$_ENV

Variables provided to the script via the environment. Analogous to the old `$HTTP_ENV_VARS` array (which is still available, but deprecated).

$_REQUEST

Variables provided to the script via any user input mechanism, and which therefore cannot be trusted. Note: when running on the command line, this will *not* include the `argv` and `argc` entries; these are present in the `$_SERVER` array. The presence and order of variable inclusion in this array is defined according to the variables_order configuration directive. This array has no direct analogue in versions of PHP prior to 4.1.0.

$_SESSION

Variables which are currently registered to a script's session. Analogous to the old `$HTTP_SESSION_VARS` array (which is still available, but deprecated). See the Session handling functions section for more information.

Variable scope

The scope of a variable is the context within which it is defined. For the most part all PHP variables only have a single scope. This single scope spans included and required files as well. For example:

```
$a = 1;
include "b.inc";
```

Here the `$a` variable will be available within the included `b.inc` script. However, within user-defined functions a local function scope is introduced. Any variable used inside a function is by default limited to the local function scope. For example:

```
$a = 1; /* global scope */

function Test()
{
    echo $a; /* reference to local scope variable */
}

Test();
```

This script will not produce any output because the echo statement refers to a local version of the `$a` variable, and it has not been assigned a value within this scope. You may notice that this is a little bit different from the C language in that global variables in C are automatically available to functions unless specifically overridden by a local definition. This can cause some problems in that people may inadvertently change a global variable. In PHP global variables must be declared global inside a function if they are going to be used in that function. An example:

```
$a = 1;
$b = 2;

function Sum()
{
    global $a, $b;

    $b = $a + $b;
}

Sum();
echo $b;
```

The above script will output "3". By declaring `$a` and `$b` global within the function, all references to either variable will refer to the global version. There is no limit to the number of global variables that can be manipulated by a function.

A second way to access variables from the global scope is to use the special PHP-defined `$GLOBALS` array. The previous example can be rewritten as:

```
$a = 1;
$b = 2;

function Sum()
{
    $GLOBALS["b"] = $GLOBALS["a"] + $GLOBALS["b"];
}

Sum();
```

```
echo $b;
```

The `$GLOBALS` array is an associative array with the name of the global variable being the key and the contents of that variable being the value of the array element.

Another important feature of variable scoping is the *static* variable. A static variable exists only in a local function scope, but it does not lose its value when program execution leaves this scope. Consider the following example:

```
function Test ()
{
    $a = 0;
    echo $a;
    $a++;
}
```

This function is quite useless since every time it is called it sets `$a` to `0` and prints "0". The `$a++` which increments the variable serves no purpose since as soon as the function exits the `$a` variable disappears. To make a useful counting function which will not lose track of the current count, the `$a` variable is declared static:

```
function Test()
{
    static $a = 0;
    echo $a;
    $a++;
}
```

Now, every time the Test() function is called it will print the value of `$a` and increment it.

Static variables also provide one way to deal with recursive functions. A recursive function is one which calls itself. Care must be taken when writing a recursive function because it is possible to make it recurse indefinitely. You must make sure you have an adequate way of terminating the recursion. The following simple function recursively counts to 10, using the static variable `$count` to know when to stop:

```
function Test()
{
    static $count = 0;

    $count++;
    echo $count;
    if ($count < 10) {
        Test ();
    }
    $count--;
}
```

Variable variables

Sometimes it is convenient to be able to have variable variable names. That is, a variable name which can be set and used dynamically. A normal variable is set with a statement such as:

```
$a = "hello";
```

A variable variable takes the value of a variable and treats that as the name of a variable. In the above example, *hello*, can be used as the name of a variable by using two dollar signs. i.e.

```
$$a = "world";
```

At this point two variables have been defined and stored in the PHP symbol tree: `$a` with contents "hello" and `$hello` with contents "world". Therefore, this statement:

```
echo "$a ${$a}";
```

produces the exact same output as:

```
echo "$a $hello";
```

i.e. they both produce: `hello world`.

In order to use variable variables with arrays, you have to resolve an ambiguity problem. That is, if you write `$$a[1]` then the parser needs to know if you meant to use `$a[1]` as a variable, or if you wanted `$$a` as the variable and then the [1] index from that variable. The syntax for resolving this ambiguity is: `${$a[1]}` for the first case and `${$a}[1]` for the second.

Please note that variable variables cannot be used with PHP's new superglobals. This means you cannot do things like `${$_GET}`. If you are looking for a way to handle availability of superglobals and the old HTTP_*_VARS, you might want to try referencing them.

Variables from outside PHP

HTML Forms (GET and POST)

When a form is submitted to a PHP script, any variables from that form will be automatically made available to the script by PHP. If the track_vars configuration option is turned on, then these variables will be located in the associative arrays `$HTTP_POST_VARS`, `$HTTP_GET_VARS`, and/or `$HTTP_POST_FILES`, according to the source of the variable in question.

For more information on these variables, please read Predefined variables.

Esempio 7-1. Simple form variable

```
<form action="foo.php" method="post">
    Name: <input type="text" name="username"><br>
    <input type="submit">
</form>
```

When the above form is submitted, the value from the text input will be available in `$HTTP_POST_VARS['username']`. If the register_globals configuration directive is turned on, then the variable will also be available as `$username` in the global scope.

> **Nota:** The magic_quotes_gpc configuration directive affects Get, Post and Cookie values. If turned on, value (It's "PHP!") will automagically become (It\'s \"PHP!\"). Escaping is needed for DB insertion. Also see addslashes(), stripslashes() and magic_quotes_sybase.

PHP also understands arrays in the context of form variables (see the related faq). You may, for example, group related variables together, or use this feature to retrieve values from a multiple select input:

Esempio 7-2. More complex form variables

```
<form action="array.php" method="post">
    Name: <input type="text" name="personal[name]"><br>
    Email: <input type="text" name="personal[email]"><br>
    Beer: <br>
    <select multiple name="beer[]">
        <option value="warthog">Warthog
        <option value="guinness">Guinness
        <option value="stuttgarter">Stuttgarter Schwabenbräu
        </select>
    <input type="submit">
</form>
```

In PHP 3, the array form variable usage is limited to single-dimensional arrays. In PHP 4, no such restriction applies.

IMAGE SUBMIT variable names

When submitting a form, it is possible to use an image instead of the standard submit button with a tag like:

```
<input type="image" src="image.gif" name="sub">
```

When the user clicks somewhere on the image, the accompanying form will be transmitted to the server with two additional variables, sub_x and sub_y. These contain the coordinates of the user click within the image. The experienced may note that the actual variable names sent by the browser contains a period rather than an underscore, but PHP converts the period to an underscore automatically.

HTTP Cookies

PHP transparently supports HTTP cookies as defined by Netscape's Spec (http://www.netscape.com/newsref/std/cookie_spec.html). Cookies are a mechanism for storing data in the remote browser and thus tracking or identifying return users. You can set cookies using the setcookie() function. Cookies are part of the HTTP header, so the SetCookie function must be called before any output is sent to the browser. This is the same restriction as for the header() function. Any cookies sent to you from the client will automatically be turned into a PHP variable just like GET and POST method data.

If you wish to assign multiple values to a single cookie, just add *[]* to the cookie name. For example:

```
setcookie("MyCookie[]", "Testing", time()+3600);
```

Note that a cookie will replace a previous cookie by the same name in your browser unless the path or domain is different. So, for a shopping cart application you may want to keep a counter and pass this along. i.e.

Esempio 7-3. SetCookie Example

```
$Count++;
setcookie("Count", $Count, time()+3600);
setcookie("Cart[$Count]", $item, time()+3600);
```

Environment variables

PHP automatically makes environment variables available as normal PHP variables.

```
echo $HOME;  /* Shows the HOME environment variable, if set. */
```

Since information coming in via GET, POST and Cookie mechanisms also automatically create PHP variables, it is sometimes best to explicitly read a variable from the environment in order to make sure that you are getting the right version. The getenv() function can be used for this. You can also set an environment variable with the putenv() function.

Dots in incoming variable names

Typically, PHP does not alter the names of variables when they are passed into a script. However, it should be noted that the dot (period, full stop) is not a valid character in a PHP variable name. For the reason, look at it:

```
$varname.ext;  /* invalid variable name */
```

Now, what the parser sees is a variable named `$varname`, followed by the string concatenation operator, followed by the barestring (i.e. unquoted string which doesn't match any known key or reserved words) 'ext'. Obviously, this doesn't have the intended result.

For this reason, it is important to note that PHP will automatically replace any dots in incoming variable names with underscores.

Determining variable types

Because PHP determines the types of variables and converts them (generally) as needed, it is not always obvious what type a given variable is at any one time. PHP includes several functions which find out what type a variable is. They are gettype(), is_array(), is_float(), is_int(), is_object(), and is_string().

Capitolo 8. Costanti

Una costante è un identificatore (nome) per un valore. Come si può intuire, tale valore non può cambiare durante l'esecuzione dello script (le costanti `__FILE__` e `__LINE__` sono l'unica eccezione). Una costante è "case-sensitive" per default. È convenzione comune che i nomi di costante siano sempre maiuscoli.

In PHP il nome di una costante segue le regole di qualsiasi "etichetta". Un nome di costante valido inizia con una lettera o underscore, seguita da un numero qualsiasi di caratteri alfanumerici o underscore. L'espressione regolare che esprime questa convenzione è: `[a-zA-Z_\x7f-\xff][a-zA-Z0-9_\x7f-\xff]*`

> **Nota:** In questo contesto una lettera è a-z, A-Z e i caratteri ASCII dal 127 al 255 (0x7f-0xff).

Una costante è sempre globale.

Sintassi

È possibile definire una variabile utilizzando la funzione define(). Una volta definita una variabile, non è possibile cambiarne il valore o eliminarla.

Le costanti possono contenere solo dati di tipo scalare (boolean, integer, double and string).

Per ottenere il valore di una costante è sufficiente specificarne il nome. A differenza delle variabili, *non* è necessario anteporre il simbolo `$` al nome di una variabile. Si può anche utilizzare la funzione constant(), per leggere il valore di una costante, nel caso in cui se ne ottenga dinamicamente il nome. Si utilizzi get_defined_constants() per ottenere una lista delle variabili definite.

> **Nota:** Costanti e variabili (globali) si trovano in un "namespace" differente. Questo implica che generalmente `TRUE` e `$TRUE` sono differenti.

Se si utilizza il nome di una costante che non è definita, PHP assume che detto valore sia il nome della costante stessa. Quando ciò accade PHP lo seganala il problema con con un notice. Per sapere se una costante è definita, si può utilizzare la funzione defined().

Di seguito sono riportate le principali differenze dalle variabili:

- Le costanti non richiedono il simbolo di dollaro (`$`) anteposto al nome;
- Le costanti si possono definire e utilizzare in qualsiasi contesto indipendentemente dalle regole dello spazio dei nomi;
- Le costanti non si possono ridefinire o eliminare una volta che siano state definite; e
- Le costanti possono contenere solo valori scalari.

Esempio 8-1. Definizione di costanti

```
<?php
define("COSTANTE", "Ciao mondo.");
echo COSTANTE; // stampa "Ciao mondo."
echo Costante; // stampa "Costante" e genera una notice.
```

```
?>
```

Costanti predefinite

Le costanti predefinite (sempre disponibili) sono:

__FILE__ (case-insensitive)

Il nome del file dello script processato. Se usato in un file che è stato incluso tramite le funzioni include e require, allora corrisponde al nome del file incluso e non al nome del file genitore.

__LINE__ (case-insensitive)

Il numero di linea all'interno del file processato. Se usato in un file che è stato incluso tramite le funzioni include e require, allora corrisponde alla posizione nel file incluso.

PHP_VERSION

La rappresentazione in formato stringa della versione del motore PHP attualmente in uso: per esempio '4.1.0'.

PHP_OS

Il nome del sistema operativo sul quale il motore PHP è in esecuzione. Valori possibili possono essere: "AIX", "Darwin" (MacOS), "Linux", "SunOS", "WIN32", "WINNT". Nota: potrebbero essere presenti anche altri valori.

`TRUE` (case-insensitive)

Un valore `TRUE` (vedere tipo boolean).

`FALSE` (case-insensitive)

Un valore `FALSE` (vedere tipo boolean).

`NULL` (case-insensitive)

Un valore `NULL` (vedere tipo boolean).

E_ERROR

Denota un errore differente da un errore di parsing. Non è recuperabile.

E_WARNING

Denota una condizione dove il PHP sa che c'è qualcosa di sbagliato, ma continua comunque l'esecuzione; Queste possono essere gestite dallo script stesso. Un esempio può essere una regexp non valida in ereg().

E_PARSE

Il parser ha incontrato in uno script una sintassi non valida. L'errore non è recuperabile.

E_NOTICE

È accaduto qualcosa che può o può non essere un errore. L'esecuzione continua. Esempi includono: usare una stringa non racchiusa da virgolette come indice di un array o accedere ad una variabile che non è stata definita.

E_ALL

Tutte le costanti E_* racchiuse in una sola. Se usata assieme a error_reporting(), farà si che tutti gli errori rilevati dal PHP siano comunicati.

Le costanti `E_*` sono tipicamente usate in congiunzione alla funzione error_reporting() per impostare il livello di errori da comunicare. Vedere tutte queste costanti in Gestione errori.

Esempio 8-2. Uso di __FILE__ e __LINE__

```
<?php
function comunica_errore($file, $linea, $messaggio)
{
    echo "È avvenuto un errore in $file alla linea $linea: $messaggio.";
}

report_error(__FILE__, __LINE__, "Qualcosa è andato storto!");
?>
```

Capitolo 9. Expressions

Expressions are the most important building stones of PHP. In PHP, almost anything you write is an expression. The simplest yet most accurate way to define an expression is "anything that has a value".

The most basic forms of expressions are constants and variables. When you type "$a = 5", you’re assigning ’5’ into $a. ’5’, obviously, has the value 5, or in other words ’5’ is an expression with the value of 5 (in this case, ’5’ is an integer constant).

After this assignment, you’d expect $a’s value to be 5 as well, so if you wrote $b = $a, you’d expect it to behave just as if you wrote $b = 5. In other words, $a is an expression with the value of 5 as well. If everything works right, this is exactly what will happen.

Slightly more complex examples for expressions are functions. For instance, consider the following function:

```
function foo ()
{
    return 5;
}
```

Assuming you’re familiar with the concept of functions (if you’re not, take a look at the chapter about functions), you’d assume that typing `$c = foo()` is essentially just like writing `$c = 5`, and you’re right. Functions are expressions with the value of their return value. Since foo() returns 5, the value of the expression ’foo()’ is 5. Usually functions don’t just return a static value but compute something.

Of course, values in PHP don’t have to be integers, and very often they aren’t. PHP supports three scalar value types: integer values, floating point values and string values (scalar values are values that you can’t ’break’ into smaller pieces, unlike arrays, for instance). PHP also supports two composite (non-scalar) types: arrays and objects. Each of these value types can be assigned into variables or returned from functions.

So far, users of PHP/FI 2 shouldn’t feel any change. However, PHP takes expressions much further, in the same way many other languages do. PHP is an expression-oriented language, in the sense that almost everything is an expression. Consider the example we’ve already dealt with, ’$a = 5’. It’s easy to see that there are two values involved here, the value of the integer constant ’5’, and the value of $a which is being updated to 5 as well. But the truth is that there’s one additional value involved here, and that’s the value of the assignment itself. The assignment itself evaluates to the assigned value, in this case 5. In practice, it means that ’$a = 5’, regardless of what it does, is an expression with the value 5. Thus, writing something like ’$b = ($a = 5)’ is like writing ’$a = 5; $b = 5;’ (a semicolon marks the end of a statement). Since assignments are parsed in a right to left order, you can also write ’$b = $a = 5’.

Another good example of expression orientation is pre- and post-increment and decrement. Users of PHP/FI 2 and many other languages may be familiar with the notation of variable++ and variable--. These are increment and decrement operators. In PHP/FI 2, the statement ’$a++’ has no value (is not an expression), and thus you can’t assign it or use it in any way. PHP enhances the increment/decrement capabilities by making these expressions as well, like in C. In PHP, like in C, there are two types of increment - pre-increment and post-increment. Both pre-increment and post-increment essentially increment the variable, and the effect on the variable is idential. The difference is with the value of the increment expression. Pre-increment, which is written ’++$variable’, evaluates to the incremented value (PHP increments the variable before reading its value, thus the name ’pre-increment’). Post-increment, which is written ’$variable++’ evaluates to

the original value of $variable, before it was incremented (PHP increments the variable after reading its value, thus the name ’post-increment’).

A very common type of expressions are comparison expressions. These expressions evaluate to either 0 or 1, meaning `FALSE` or `TRUE` (respectively). PHP supports > (bigger than), >= (bigger than or equal to), == (equal), != (not equal), < (smaller than) and <= (smaller than or equal to). These expressions are most commonly used inside conditional execution, such as `if` statements.

The last example of expressions we’ll deal with here is combined operator-assignment expressions. You already know that if you want to increment $a by 1, you can simply write ’$a++’ or ’++$a’. But what if you want to add more than one to it, for instance 3? You could write ’$a++’ multiple times, but this is obviously not a very efficient or comfortable way. A much more common practice is to write ’$a = $a + 3’. ’$a + 3’ evaluates to the value of $a plus 3, and is assigned back into $a, which results in incrementing $a by 3. In PHP, as in several other languages like C, you can write this in a shorter way, which with time would become clearer and quicker to understand as well. Adding 3 to the current value of $a can be written ’$a += 3’. This means exactly "take the value of $a, add 3 to it, and assign it back into $a". In addition to being shorter and clearer, this also results in faster execution. The value of ’$a += 3’, like the value of a regular assignment, is the assigned value. Notice that it is NOT 3, but the combined value of $a plus 3 (this is the value that’s assigned into $a). Any two-place operator can be used in this operator-assignment mode, for example ’$a -= 5’ (subtract 5 from the value of $a), ’$b *= 7’ (multiply the value of $b by 7), etc.

There is one more expression that may seem odd if you haven’t seen it in other languages, the ternary conditional operator:

```
$first ? $second : $third
```

If the value of the first subexpression is `TRUE` (non-zero), then the second subexpression is evaluated, and that is the result of the conditional expression. Otherwise, the third subexpression is evaluated, and that is the value.

The following example should help you understand pre- and post-increment and expressions in general a bit better:

```
function double($i)
{
    return $i*2;
}
$b = $a = 5;        /* assign the value five into the variable $a and $b */
$c = $a++;          /* post-increment, assign original value of $a
                       (5) to $c */
$e = $d = ++$b;     /* pre-increment, assign the incremented value of
                       $b (6) to $d and $e */

/* at this point, both $d and $e are equal to 6 */

$f = double($d++);  /* assign twice the value of $d <emphasis>before</emphasis>
                       the increment, 2*6 = 12 to $f */
$g = double(++$e);  /* assign twice the value of $e <emphasis>after</emphasis>
                       the increment, 2*7 = 14 to $g */
$h = $g += 10;      /* first, $g is incremented by 10 and ends with the
                       value of 24. the value of the assignment (24) is
```

```
                         then assigned into $h, and $h ends with the value
                         of 24 as well. */
```

In the beginning of the chapter we said that we’ll be describing the various statement types, and as promised, expressions can be statements. However, not every expression is a statement. In this case, a statement has the form of ’expr’ ’;’ that is, an expression followed by a semicolon. In ’$b=$a=5;’, $a=5 is a valid expression, but it’s not a statement by itself. ’$b=$a=5;’ however is a valid statement.

One last thing worth mentioning is the truth value of expressions. In many events, mainly in conditional execution and loops, you’re not interested in the specific value of the expression, but only care about whether it means `TRUE` or `FALSE`. The constants `TRUE` and `FALSE` (case-insensitive) are the two possible boolean values. When necessary, an expression is automatically converted to boolean. See the section about type-casting for details about how.

PHP provides a full and powerful implementation of expressions, and documenting it entirely goes beyond the scope of this manual. The above examples should give you a good idea about what expressions are and how you can construct useful expressions. Throughout the rest of this manual we’ll write *`expr`* to indicate any valid PHP expression.

Capitolo 10. Operatori

Operatori aritmetici

Ricordate l'aritmetica di base dalla scuola? Questi funzionano proprio come quelli.

Tabella 10-1. Operatori aritmetici

Esempio	Nome	Risultato
$a + $b	Addizione	La somma di $a e $b.
$a - $b	Sottrazione	La differenza di $a e $b.
$a * $b	Moltiplicazione	il prodotto di $a e $b.
$a / $b	Divisione	Quoziente di $a e $b.
$a % $b	Modulo	Il resto di $a diviso da $b.

L'operatore di divisione ("/") restituisce un valore intero (il risultato di una divisione intera) se i due operandi sono interi (o stringhe che vengono convertite a interi) e il quoziente è un intero. Se uno degli operandi (o entrambi) è un valore in virgola mobile, oppure il risultato dell'operazione è un valore non intero, viene restituito un valore in virgola mobile.

Operatori di assegnazione

L'operatore di base dell'assegnazione è "=". La vostra prima inclinazione potrebbe essere di pensare che ciò sia come "uguale a". No. Esso significa realmente che l'operando a sinistra assume il valore dell'espressione a destra (ciò significa, "assegna il valore a").

Il valore di un'espressione di assegnazione è il valore assegnato. Cioè il valore di "$a = 3" è 3. Questo vi permette di fare qualche trucchetto:

```
$a = ($b = 4) + 5; // $a è uguale a 9 ora, e $b è stato impostato a 4.
```

In aggiunta all'operatore di base dell'assegnazione, ci sono gli "operatori combinati" per tutta l'aritmetica binaria e gli operatori di stringa che vi consentono di usare un valore in un'espressione e poi impostare il suo valore al risultato di quell'espressione. Per esempio:

```
$a = 3;
$a += 5; // imposta $a a 8, come se avessimo detto: $a = $a + 5;
$b = "Ciao ";
$b .= "Lì!"; // imposta $b a "Ciao Lì!", proprio come $b = $b . "Lì!";
```

Notare che l'assegnazione copia la variabile originale alla nuova (assegnazione per valore), così i cambiamenti ad una non si verificheranno nell' altra. Ciò può anche avere rilevanza se avete bisogno di copiare qualcosa come un grande array in un ciclo molto stretto. PHP 4 supporta l'assegnazione per riferimento, usando la sintassi `$var = &$othervar;`, ma ciò non è possibile in PHP 3.

’L’assegnazione per riferimento’ vuol dire che entrambe le variabili finiscono con il puntare agli stessi dati, e nulla è copiato in nessun posto. Per ulteriori approfondimenti sui riferimenti, consultare References explained.

Operatori bitwise

Gli operatori bitwise vi permettono di alterare bit specifici in posizione on oppure off. Se entrambi i parametri di sinistra e destra sono stringhe, l’operatore bitwise opererà sui caratteri di questa stringa.

```
<?php
    echo 12 ^ 9; // L’output è ’5’

    echo "12" ^ "9"; // L’output è il carattere Backspace (ascii 8)
                     // (’1’ (ascii 49)) ^ (’9’ (ascii 57)) = #8

    echo "hallo" ^ "hello"; // L’output è il valore ascii #0 #4 #0 #0 #0
                            // ’a’ ^ ’e’ = #4
?>
```

Tabella 10-2. Operatori bitwise

Esempio	Nome	Risultato
$a & $b	And	Sono impostati ad ON i bit che sono ON sia in $a che in $b.
$a | $b	Or	Sono impostati ad ON i bit che sono ON in $a oppure in $b.
$a ^ $b	Xor	Sono impostati ad ON i bit che sono ON in $a oppure in $b na non quelli che sono entrambi ON.
~ $a	Not	Sono impostati ad ON i bit che sono OFF in $a, e viceversa.
$a << $b	Shift left	Sposta i bit di $a a sinistra di $b passi (ogni passo significa "moltiplica per due")
$a >> $b	Shift right	Sposta i bit di $a a destra di $b passi (ogni passo significa "dividi per due")

Operatori di confronto

Gli operatori di confronto, come suggerisce il loro nome, permettono di confrontare due valori.

Tabella 10-3. Operatori di confronto

Esempio	Nome	Risultato
$a == $b	Uguale	TRUE se $a è uguale a $b.
$a === $b	Identico	TRUE se $a è uguale a $b, ed essi sono dello stesso tipo. (Solo PHP 4)
$a != $b	Diversi	TRUE se $a è diverso da $b.
$a <> $b	Diversi	TRUE se $a è diverso da $b.
$a !== $b	Non identici	TRUE se $a è diverso da $b, o se essi non sono dello stesso tipo. (Solo PHP 4)
$a < $b	Minore	TRUE se $a è strettamente minore di $b.
$a > $b	Maggiore	TRUE se $a è strettamente maggiore di $b.
$a <= $b	Minore o uguale	TRUE se $a è minore o uguale a $b.
$a >= $b	Maggiore o uguale	TRUE se $a è maggiore o uguale a $b.

Un altro operatore condizionale è l'operatore "?:" (o trinario), che opera come in C e molti altri linguaggi.

```
(espressione1) ? (espressione2) : (espressione3);
```

Questa espressione vale *espressione2* se *espressione1* è TRUE, e *espressione3* se *espressione1* è FALSE.

Operatori di controllo errori

PHP supporta un operatore di controllo dell'errore: il carattere at (@). Quando prefisso ad una espressione in PHP, qualunque messaggio di errore che potesse essere generato da quella espressione sarà ignorato.

Se la caratteristica track_errors è abilitata, qualsiasi messaggio di errore generato dall'espressione sarà salvato nella variabile globale $php_errormsg. Questa variabile sarà sovrascritta ad ogni errore, così controllatela subito se volete usarla.

```
<?php
/* Errore di file intenzionale */
$my_file = @file ('file_inesistente') or
    die ("Apertura del file fallita: l'errore è '$php_errormsg'");

// questo funziona per qualsiasi espressione, non solo funzioni:
```

```
$value = @$cache[$key];
// non verrà generata una notifica se l'indice $key non esiste.

?>
```

Nota: L'operatore @ funziona solo sulle espressioni. Una semplice regola di thumb è: se potete prendere il valore di qualcosa, potete anteporre ad esso l'operatore @. Per esempio, potete anteporre esso a variabili, funzioni e chiamate ad include(), costanti, e così via. non potete anteporre esso a definizioni di funzioni o classi, o strutture condizionali come `if` e `foreach`, e così via.

Vedere anche error_reporting().

Attenzione

Attualmente il prefisso operatore di controllo dell'errore "@" disabiliterà la restituzione di errori per errori critici che interromperanno l'esecuzione dello script. Tra le altre cose, questo significa che se state usando "@" per sopprimere errori da una certa funzione ed essa non è disponibile oppure è stata scritta male, lo script terminerà senza dare indicazioni sul perché.

Operatori di esecuzione

PHP supporta un operatore di esecuzione: backticks (“). Notare che quelli non sono apostrofi! PHP cercherà di eseguire il contenuto dei backticks come comando di shell; sarà restituito l'output (i.e., non sarà semplicemente esportato come output; può essere assegnato ad una variabile).

```
$output = `ls -al`;
echo "<pre>$output</pre>";
```

Nota: L'operatore backtick è disabilitato quando è abilitata safe mode oppure quando è disabilitata shell_exec().

Vedere anche escapeshellcmd(), exec(), passthru(), popen(), shell_exec() e system().

Operatori di incremento/decremento

PHP supporta lo stile C degli operatori di pre- e post-incremento e decremento.

Tabella 10-4. Operatori di incremento/decremento

Esempio	Nome	Effetto
++$a	Pre-incremento	Incrementa $a di una unità, inoltre restituisce $a.
$a++	Post-incremento	Restituisce $a, inoltre incrementa $a di una unità.
--$a	Pre-decremento	Decrementa $a di una unità, inoltre restituisce $a.
$a--	Post-decremento	Restituisce $a, inoltre decrementa $a di una unità.

Qui c’è un semplice script di esempio:

```
<?php
echo "<h3&gt;Post-incremento</h3&gt;";
$a = 5;
echo "Dovrebbe essere 5: " . $a++ . "<br>\n";
echo "Dovrebbe essere 6: " . $a . "<br>\n";

echo "<h3>Pre-incremento</h3>";
$a = 5;
echo "Dovrebbe essere 6: " . ++$a . "<br>\n";
echo "Dovrebbe essere 6: " . $a . "<br>\n";

echo "<h3>Post-decremento</h3>";
$a = 5;
echo "Dovrebbe essere 5: " . $a-- . "<br>\n";
echo "Dovrebbe essere 4: " . $a . "<br>\n";

echo "<h3>Pre-decremento</h3>";
$a = 5;
echo "Dovrebbe essere 4: " . --$a . "<br>\n";
echo "Dovrebbe essere 4: " . $a . "<br>\n";
?>
```

Operatori logici

Tabella 10-5. Operatori logici

Esempio	Nome	Risultato
$a and $b	And	`TRUE` se entrambi $a e $b sono `TRUE`.
$a or $b	Or	`TRUE` se uno tra $a o $b è `TRUE`.

<table>
<tr><th>Esempio</th><th>Nome</th><th>Risultato</th></tr>
<tr><td>$a xor $b</td><td>Xor</td><td>TRUE se uno tra $a o $b è TRUE, ma non entrambi.</td></tr>
<tr><td>! $a</td><td>Not</td><td>TRUE se $a non è TRUE.</td></tr>
<tr><td>$a && $b</td><td>And</td><td>TRUE se entrambi $a e $b sono TRUE.</td></tr>
<tr><td>$a || $b</td><td>Or</td><td>TRUE se uno tra $a o $b è TRUE.</td></tr>
</table>

La ragione per le due differenti variazioni degli operatori "and" e "or" è che essi operano con differenti precedenze. (Vedere Precedenza degli operatori.)

Precedenza degli operatori

La precedenza di un operatore specifica come esso tenga legate assieme "strettamente" due espressioni. Per esempio, nell'espressione `1 + 5 * 3`, la risposta è `16` e non `18` perché l'operatore di moltiplicazione ("*") ha una precedenza più alta rispetto all'operatore di addizione ("+"). Le parentesi possono essere usate per forzare la precedenza, se necessario. Per esempio: `(1 + 5) * 3` viene valutata `18`.

La seguente tabella fornisce una lista della precedenza degli operatori con gli operatori a più bassa precedenza listati prima.

Tabella 10-6. Precedenza degli operatori

<table>
<tr><th>Associatività</th><th>Operatori</th></tr>
<tr><td>sinistra</td><td>,</td></tr>
<tr><td>sinistra</td><td>or</td></tr>
<tr><td>sinistra</td><td>xor</td></tr>
<tr><td>sinistra</td><td>and</td></tr>
<tr><td>destra</td><td>print</td></tr>
<tr><td>sinistra</td><td>= += -= *= /= .= %= &= |= ^= ~= <<= >>=</td></tr>
<tr><td>sinistra</td><td>? :</td></tr>
<tr><td>sinistra</td><td>||</td></tr>
<tr><td>sinistra</td><td>&&</td></tr>
<tr><td>sinistra</td><td>|</td></tr>
<tr><td>sinistra</td><td>^</td></tr>
<tr><td>sinistra</td><td>&</td></tr>
<tr><td>non associativi</td><td>== != === !==</td></tr>
<tr><td>non associativi</td><td>< <= > >=</td></tr>
<tr><td>sinistra</td><td><< >></td></tr>
<tr><td>sinistra</td><td>+ - .</td></tr>
<tr><td>sinistra</td><td>* / %</td></tr>
<tr><td>destra</td><td>! ~ ++ -- (int) (double) (string) (array) (object) @</td></tr>
<tr><td>destra</td><td>[</td></tr>
<tr><td>non associativi</td><td>new</td></tr>
</table>

Operatori di stringa

Ci sono due operatori di stringa. Il primo è l'operatore di concatenazione ('.'), che restituisce la concatenazione dei suoi argomenti a destra e a sinistra. Il secondo è l'operatore di assegnazione concatenata ('.='), che aggiunge alla fine dell'argomento sul lato destro l'argomento sul lato sinistro. Per favore consultare Operatori di assegnamento per maggiori informazioni.

```
$a = "Ciao ";
$b = $a . "Mondo!"; // ora $b contiene "Ciao Mondo!"

$a = "Ciao ";
$a .= "Mondo!";     // ora $a contiene "Ciao Mondo!"
```

Capitolo 11. Strutture di controllo

Qualsiasi script PHP è costituito da una serie di istruzioni. Una istruzione può essere un'assegnazione, una chiamata di funzione, un loop, una istruzione condizionale che non fa nulla (istruzione vuota). Le istruzioni terminano con un punto e virgola. Inoltre, le istruzioni si possono raggruppare in blocchi di istruzioni racchiudendole tra parentesi graffa. Un gruppo di istruzioni è, a sua volta, un'istruzione. Il presente capitolo descrive i differenti tipi di istruzioni.

`if`

Il costrutto `if` è una delle più importanti caratteristiche di qualsiasi linguaggio, incluso PHP. Permette l'esecuzione condizionata di frammenti di codice. La struttura di controllo `if` di PHP è simile a quella del linguaggio C:

```
if (espressione)
    istruzione
```

Come descritto nella sezione sulle espressioni, *`espressione`* restiruirà il suo valore booleano. Se *`espressione`* vale `TRUE`, PHP eseguirà *`istruzione`*, e se essa vale `FALSE` - la ignorerà. Più informazioni riguardo i valori valutati `FALSE` possono essere trovati nella sezione 'Conversione in booleano' .

L'esempio che segue visualizzerà `a è maggiore di b` se *`$a`* sarà maggiore di *`$b`*:

```
if ($a > $b)
    print "a è maggiore di b";
```

Spesso sarà necessario eseguire più di una istruzione condizionale. Naturalmente non è necessario, utilizzare una singola clausola `if` per ciascuna istruzione. Si possono raggruppare diverse istruzioni in un singolo gruppo di istruzioni. Per esempio, il codice che segue visualizzerà `a è maggiore di b` se *`$a`* è maggiore di *`$b`*, e successivamente assegnerà il valore della variabile *`$a`* alla variabile *`$b`*:

```
if ($a > $b) {
    print "a è maggiore di b";
    $b = $a;
}
```

Si possono annidare indefinitamente istruzioni `if`, la qual cosa fornisce piena flessibilità per l'esecuzione di istruzioni condizionali in diversi punti del programma.

else

Spesso è necessario eseguire un'istruzione se una proposizione è vera e un'altra istruzione se la proposizione è falsa. Per questo si usa la clausola `else`. `else` estende il costrutto `if` aggiungendo la possibilità di eseguire un'istruzione se l'espressione nel ramo `if` è `FALSE`. L'esempio che segue visualizzerà `a è maggiore di b` se *$a* è maggiore di *$b* e `a NON è maggiore di b` altrimenti:

```
if ($a > $b) {
    print "a è maggiore di b";
} else {
    print "a NON è maggiore di b";
}
```

Il ramo `else` viene eseguito solo se l'espressione nel ramo `if` è `FALSE`, e, nel caso ci fossero delle clausole `elseif`, solamente se le espressioni in esse contenute fossero anch'esse `FALSE` (vedere elseif).

elseif

`elseif`, come è facile intuire, è una combinazione di `if` ed `else`. Analogamente ad `else`, estende `if` aggiungendo la possibilità di eseguire un'altra istruzione nel caso in cui l'espressione contenuta nel ramo `if` sia `FALSE`. Però, a differenza di `else`, si eseguirà l'istruzione alternativa solamente se l'espressione contenuta nel ramo `elseif` sarà `TRUE`. L'esempio che segue, visualizzerà `a è maggiore di b`, `a è uguale a b` oppure `a è minore di b`:

```
if ($a > $b) {
    print "a è maggiore di b";
} elseif ($a == $b) {
    print "a è uguale a b";
} else {
    print "a è minore di b";
}
```

Nel medesimo blocco `if` possono essere presenti più di una clausola `elseif`. Verrà eseguita l'istruzione del primo ramo `elseif` la cui espressione sia `TRUE`. In PHP è possibile scrivere 'else if' (due parole) e il significato sarà lo stesso di 'elseif' (una sola parola). Il significato sintattico è leggermente differente (se si ha familiarità con il linguaggio C, esso ha lo stesso comportamento) però al lato pratico l'effetto è il medesimo.

L'istruzione di un ramo `elseif` verrà eseguita solo se l'espressione del ramo `if` e le espressioni dei rami `elseif` precedenti sono `FALSE`, e se l'espressione del ramo `elseif` è `TRUE`.

Sintassi alternativa per le strutture di controllo

PHP offre una sintassi alternativa per alcune delle sue strutture di controllo; vale a dire, `if`, `while`, `for`, `foreach` e `switch`. Fondamentalmente la sintassi alternativa consiste nel sostituire la prima parentesi graffa con il carattere "duepunti" (:) e la seconda parentesi graffa con `endif;`, `endwhile;`, `endfor;`, `endforeach;`, oppure `endswitch;`, rispettivamente.

```
<?php if ($a == 5): ?>
a è uguale a 5
<?php endif; ?>
```

Nell'esempio precedente, il blocco HTML "a è uguale a 5" è incluso nel ramo `if` scritto utilizzando la sintassi alternativa. Il blocco HTML verrà visualizzato solamente se $a è uguale a 5.

La sintassi alternativa si applica anche ad `else` ed `elseif`. Nell'esempio che segue si mostra come utilizzare la sintassi alternativa nel caso di un `if` con `elseif` ed `else`:

```
if ($a == 5):
    print "a è uguale a 5";
    print "...";
elseif ($a == 6):
    print "a è uguale a 6";
    print "!!!";
else:
    print "a non è nè 5 nè 6";
endif;
```

Vedere anche while, for, e if per ulteriori esempi.

`while`

Il ciclo `while` è la forma di ciclo più semplice tra quelle possibili in PHP. Si comporta come la sua controparte nel linguaggio C. La forma di base di un ciclo `while` è la seguente:

```
while (espressione) istruzione
```

Il significato di un ciclo `while` è semplice. Istruisce l'interprete PHP perchè esegua l'istruzione (o le istruzioni) in esso racchiuse, ripetutamente, fintanto che l'espressione contenuta nella clausola `while` ha valore `TRUE`. Il valore dell'espressione viene verificato ogni volta che il ciclo si ripete

(iterazione), così che anche se il valore dell'espressione cambia durante l'esecuzione dell'istruzione, il ciclo non termina fino all'iterazione successiva. Ovviamente, se l'espressione nella clausola `while` ha valore `FALSE` dall'inizio, l'istruzione racchiusa nel blocco non verrà eseguita nemmeno una volta.

Come nel caso della struttura di controllo `if`, si possono raggruppare più istruzioni nello medesimo ciclo `while` racchiudendo le istruzioni in parentesi graffa, oppure utilizzando la sintassi alternativa:

```
while (espressione): istruzione ... endwhile;
```

Gli esempi seguenti sono identici e entrambi visualizzano i numeri da 1 a 10:

```
/* esempio 1 */

$i = 1;
while ($i <= 10) {
    print $i++;  /* Il valore visualizzato è il valore della
                    variabile $i prima dell'incremento
                    (post-incremento) */
}

/* esempio 2 */

$i = 1;
while ($i <= 10):
    print $i;
    $i++;
endwhile;
```

do..while

Il ciclo `do..while` è simile al ciclo `while`, con l'unica differenza che il valore dell'espressione viene controllato alla fine di ogni iterazione anzichè all'inizio. La differenza più importante rispetto a `while` è che la prima iterazione di un blocco `do..while` verrà sempre eseguita (il valore dell'espressione viene controllato alla fine del ciclo), mentre non sarà necessariamente eseguito in un ciclo `while` (il valore dell'espressione viene controllato all'inizio del ciclo, e se tale valore è `FALSE` dall'inizio, l'esecuzione del ciclo termina immediatamente).

È ammessa una sola sintassi per il ciclo `do..while`:

```
$i = 0;
do {
   print $i;
} while ($i>0);
```

Il ciclo precedente verrà eseguito un'unica volta, dal momento che alla prima iterazione, quando si controlla l'espressione, il suo valore sarà `FALSE` ($i non è maggiore di 0) e il ciclo di esecuzioni, termina.

Chi ha utilizzato il linguaggio C conosce probabilmente un'altro modo di utilizzare il ciclo `do..while`, che permette di terminare l'esecuzione delle istruzioni durante l'esecuzione stessa, utilizzando `do..while`(0), e usando l'istruzione `break`. Il codice che segue esemplifica questa possibilità:

```
do {
    if ($i < 5) {
        print "i non è abbastanza grande";
        break;
    }
    $i *= $factor;
    if ($i < $minimum_limit) {
        break;
    }
    print "i è ok";

     ...processa i...

} while(0);
```

Non vi preoccupate se l'esempio non è sufficientemente chiaro. Si possono scrivere ottimi programmi PHP anche senza far ricorso a questa 'possibilità'.

`for`

Il ciclo `for` è il ciclo più complesso tra quelli disponibili in PHP. Si comporta come la sua controparte nel linguaggio C. La sintassi di un clico `for` è:

```
for (espressione1; espressione2; espressione3) istruzione
```

Il valore della prima espressione (*`espressione1`*) viene verificato (eseguito) una sola volta incondizionatamente all'inizio del ciclo.

Ad ogni iterazione, si controlla il valore di *`espressione2`*. Se è `TRUE`, il ciclo prosegue e viene eseguita l'istruzione (o le istruzioni) contenuta nel blocco; se è `FALSE`, l'esecuzione del ciclo termina.

Al termine di ogni iterazione, si verifica (si esegue) il valore di *`espressione3`*.

Le due espressioni possono anche non essere presenti. Se non esiste *`espressione2`* significa che il ciclo deve essere eseguito indefinitamente (PHP considera implicitamente che il suo valore è `TRUE`, come in C). Questa possibilità in fondo non è utile come può sembrare perchè obbliga a terminare il ciclo utilizzando l'istruzione `break` invece di utilizzare le espressioni booleane del ciclo `for` .

Si considerino gli esempi seguenti. In ciascun caso si visualizzeranno i numeri da 1 a 10:

```
/* esempio 1 */

for ($i = 1; $i <= 10; $i++) {
    print $i;
}

/* esempio 2 */

for ($i = 1;;$i++) {
    if ($i > 10) {
        break;
    }
    print $i;
}

/* esempio 3 */

$i = 1;
for (;;) {
    if ($i > 10) {
        break;
    }
    print $i;
    $i++;
}

/* esempio 4 */

for ($i = 1; $i <= 10; print $i, $i++) ;
```

Naturalmente il primo esempio sembra il migliore (o forse il quarto), ma l'uso del ciclo `for` senza espressioni può essere utile in molti casi.

PHP offre una sintassi alternativa (con i "punto e virgola") per i cicli `for`.

```
for (espressione1; espressione2; espressione3): istruzione; ...; endfor;
```

Alcuni linguaggi permettono l'uso della struttura di controllo `foreach` per attraversare un array o una tabella hash. PHP 3 non premette l'uso di tale ciclo mentre PHP 4 si (vedere foreach). In PHP 3 è possibile combinare while con la funzione list() e each() per ottenere la stessa funzionalià. Si veda la documentazione di queste funzioni per ulteriori esempi.

foreach

PHP 4 (non PHP 3) permette l'uso della struttura di controllo `foreach`, alla stessa maniera del linguaggio Perl e altri. Ciò semplicemente fornisce una facile metodo per attraversare un array. Esistono due possibili notazioni sintattiche; la seconda è un'utile estensione della prima:

```
foreach(array_expression as $value) istruzione
foreach(array_expression as $key => $value) istruzione
```

La prima attraversa l'array dato da `array_expression`. Ad ogni ciclo, si assegna il valore dell'elemento corrente a `$value` e il puntatore interno avanza di una posizione (in modo tale che al ciclo successivo l'elemento corrente sarà il successivo elemento dell'array).

La seconda esegue lo stesso ciclo con la differenza che il valore dell'indice corrente viene assegnato ad ogni ciclo, alla variabile `$key`.

> **Nota:** All'inizio dell'esecuzione di un ciclo `foreach` il puntatore interno viene automaticamente posizionato nella prima posizione. Questo significa che non è necessario utilizzare la funzione reset() prima di un ciclo `foreach` .

> **Nota:** È importante notare che `foreach` opera su una copia dell'array, non sull'array stesso, pertanto il puntatore dell'array originale non viene modificato come accade utilizzando la funzione each() e le modifiche agli elementi dell'array non appaiono nell'array originale.

> **Nota:** `foreach` non offre la possibilità di annullare la generazione di messaggi d'errore utilizzando il carattere '@'.

Avete probabilmente notato che i due cicli seguenti sono identici da un punto di vista funzionale:

```
reset ($arr);
while (list(, $value) = each ($arr)) {
    echo "Valore: $value<br>\n";
}

foreach ($arr as $value) {
    echo "Valore: $value<br>\n";
}
```

Allo stesso modo i due cicli seguenti sono identici:

```
reset ($arr);
while (list($key, $value) = each ($arr)) {
    echo "Chiave: $key; Valore: $value<br>\n";
}

foreach ($arr as $key => $value) {
    echo "Chiave: $key; Valore: $value<br>\n";
}
```

Di seguito, altri esempi per mostrare possibili utilizzi:

```
/* esempio 1 foreach: solo il valore */

$a = array (1, 2, 3, 17);

foreach ($a as $v) {
   print "Valore corrente di \$a: $v.\n";
}

/* esempio 2 foreach: valore (con la chiave stampata) */

$a = array (1, 2, 3, 17);

$i = 0; /* solo per un proposito illustrativo */

foreach($a as $v) {
    print "\$a[$i] => $v.\n";
    $i++;
}

/* esempio 3 foreach: chiave e valore */

$a = array (
    "uno" => 1,
    "due" => 2,
    "tre" => 3,
    "diciassette" => 17
);

foreach($a as $k => $v) {
    print "\$a[$k] => $v.\n";
}

/* esempio 4 foreach: array multidimensionali */

$a[0][0] = "a";
$a[0][1] = "b";
$a[1][0] = "y";
$a[1][1] = "z";
```

```
foreach($a as $v1) {
    foreach ($v1 as $v2) {
        print "$v2\n";
    }
}

/* esempio 5 foreach: array dinamici */

foreach(array(1, 2, 3, 4, 5) as $v) {
    print "$v\n";
}
```

break

`break` termina l'esecuzione di una struttura `for`, `foreach` `while`, `do..while` o `switch`.

`break` accetta un argomento opzionale che definisce, nel caso di cicli annidati, il livello del ciclo che è da interrompere.

```
$arr = array ('uno', 'due', 'tre', 'quattro', 'stop', 'cinque');
while (list (, $val) = each ($arr)) {
    if ($val == 'stop') {
        break;    /* Qui si può anche usare 'break 1;'. */
    }
    echo "$val<br>\n";
}

/* Uso dell'argomento opzionale. */

$i = 0;
while (++$i) {
    switch ($i) {
    case 5:
        echo "At 5<br>\n";
        break 1;  /* Interrompe solo awitch. */
    case 10:
        echo "At 10; quitting<br>\n";
        break 2;  /* Interrompe switch e while. */
    default:
        break;
    }
}
```

continue

`continue` si utilizza per interrompere l'esecuzione del ciclo corrente e continuare con l'esecuzione all'inizio del ciclo successivo.

`continue` accetta un argomento numerico opzionale che definisce, nel caso di cicli annidati, il numero di cicli da interrompere e da cui iniziare l'esecuzione dell'iterazione successiva.

```
while (list ($key, $value) = each ($arr)) {
    if (!($key % 2)) { // salta odd members
        continue;
    }
    do_something_odd ($value);
}

$i = 0;
while ($i++ &lt; 5) {
    echo "Esterno<br>\n";
    while (1) {
        echo "  Centrale<br>\n";
        while (1) {
            echo "  Interno<br>\n";
            continue 3;
        }
        echo "Questo non sarà mai stampato.<br>\n";
    }
    echo "Nemmeno questo.<br>\n";
}
```

switch

`switch` è simile a una serie di if sulla stessa espressione. In molti casi può essere necessario confrontare una variabile (o espressione) con differenti valori ed eseguire un differente blocco di istruzioni a seconda del valore di detta variabile. Questo è esattamente quello che fa la struttura di controllo `switch`.

Gli esempi seguenti mostrano due maniere differenti di scrivere la stessa cosa, uno utilizzando una serie di `if`, l'altro utilizzando `switch` :

```
if ($i == 0) {
    print "i è uguale a 0";
}
if ($i == 1) {
    print "i è uguale a 1";
}
if ($i == 2) {
    print "i è uguale a 2";
}
```

```
switch ($i) {
    case 0:
        print "i è uguale a 0";
        break;
    case 1:
        print "i è uguale a 1";
        break;
    case 2:
        print "i è uguale a 2";
        break;
}
```

È importante comprendere esattamente come viene eseguita la clausola `switch` per evitare errori. Un'istruzione `switch` esegue linea dopo linea le istruzioni in essa contenuta. All'inizio non viene eseguito alcun codice. Solamente quando incontra una clausola `case` il cui valore è uguale al valore della viariabile, PHP inizia ad eseguire le istruzioni contenute nel blocco `case`. PHP continua l'esecuzione delle istruzioni fino alla termine del blocco `switch`, o quando incontra un'istruzione `break`. Se non esiste alcuna istruzione `break` al termine di un blocco case PHP continuerà l'esecuzione delle istruzioni del blocco case successivo. Per esempio:

```
switch ($i) {
    case 0:
        print "i è uguale a 0";
    case 1:
        print "i è uguale a 1";
    case 2:
        print "i è uguale a 2";
}
```

In questo caso se $i è uguale a 0, PHP eseguirà tutte le istruzioni contenute nei blocchi case. Se $i è uguale a 1, PHP eseguirà le istruzioni degli ultimi due blocchi case e solamente se $i è uguale a 2 otterremo il risultato voluto e si visualizzerà solo '$i è uguale a 2'. Pertanto è importante non dimenticare l'istruzione `break` (anche se in alcuni casi potrà essere necessario non utilizzarla).

In un'istruzione `switch`, la condizione in parentesi viene valutata una sola volta e il risultato viene confrontato con ciascun ramo `case`. Utilizzando `elseif`, la condizione viene valutata una seconda volta. Se tale condizione è più complessa di un semplice confronto e/o è in un ciclo piuttosto pesante, l'uso di `switch` dovrebbe garantire un minor tempo di esecuzione.

Un blocco case può anche non contenere istruzioni, nel qual caso il controllo passa semplicemente al successivo blocco case.

```
switch ($i) {
    case 0:
    case 1:
    case 2:
        print "i è minore di 3 ma non negativo";
```

```
        break;
    case 3:
        print "i è 3";
}
```

Un blocco case speciale è il il blocco case di default. Uguaglia tutte le condizioni non uguagliate nei blocchi case precedenti e dev'essere l'ultimo blocco `case`. Per esempio:

```
switch ($i) {
    case 0:
        print "i è uguale a 0";
        break;
    case 1:
        print "i è uguale a 1";
        break;
    case 2:
        print "i è uguale a 2";
        break;
    default:
        print "i è diverso da 0, 1 o 2";
}
```

L'espressione in un ramo `case` può essere qualsiasi espressione il cui valore sarà di tipo intero, decimale, numerico e stringa. Array e oggetti (objects) non sono ammessi a meno che non siano dereferenziati a un tipo di dato semplice tra quelli precedentemente elencati.

Come per altre strutture di controllo è possibile utilizzare una sintassi alternativa. Si veda Sintassi alternativa per le strutture di controllo per ulteriori esempi.

```
switch ($i):
    case 0:
        print "i è uguale a 0";
        break;
    case 1:
        print "i è uguale a 1";
        break;
    case 2:
        print "i è uguale a 2";
        break;
    default:
        print "i è diverso da 0, 1 o 2";
endswitch;
```

`declare`

Il costrutto `declare` si usa per definire direttive di esecuzione per blocchi di istruzioni. La sintassi è simile alla sintassi di altre strutture di controllo:

```
declare (direttiva) istruzione
```

La sezione `direttiva` permette di impostare il comportamento del blocco `declare` . Attualmente è riconosciuta una sola direttiva: la direttiva `ticks`. (Fare riferimento più in basso per ulteriori informazioni relative alla direttiva ticks)

Verrà eseguita la parte `istruzione` del blocco `declare` - come verrà eseguita e quali effetti collaterali emergeranno durante l'esecuzione potrà dipendere dalla direttiva impostata nel blocco `direttiva`.

Ticks

Un tick è un evento che si verifica per ogni *`N`* istruzioni di basso livello eseguite dal parser all'interno del blocco `declare`. Il valore per *`N`* viene specificato usando `ticks=N` all'interno della sezione `direttiva` del blocco `declare`.

L'evento (o gli eventi) che si verifica su ogni tick è specificato usando register_tick_function(). Vedere l'esempio più in basso per ulteriori dettagli. Notare che può verificarsi più di un evento per ogni tick.

Esempio 11-1. Segue una sezione di codice PHP

```
<pre>
<?php
// Una funzione che registra il tempo quando viene chiamata
function profile ($dump = FALSE)
{
    static $profile;

    // Restituisce il tempo immagazinato in $profile, successivamente lo cancella
    if ($dump) {
        $temp = $profile;
        unset ($profile);
        return ($temp);
    }

    $profile[] = microtime ();
}

// Imposta un tick handler
register_tick_function("profile");

// Inizializza la funzione prima del blocco declare
profile ();
```

```
// Esegue un blocco di codice, attraverso un tick ogni seconda istruzione
declare (ticks = 2) {
    for ($x = 1; $x < 50; ++$x) {
        echo similar_text (md5($x), md5($x*$x)), "&lt;br&gt;";
    }
}

// Mostra i dati immagazionati nel profilo
print_r (profile (TRUE));
?>
</pre>
```

L'esempio descrive il codice PHP all'interno del blocco 'declare', registrando il tempo in cui è stata eseguita ogni seconda istruzione di basso livello. Questa informazione può poi essere usata per trovare le aree lente all'interno di particolari segmenti di codice. Questo processo può essere ottimizzato usando altri metodi: usando i tick è più conveniente e facile da implementare.

I tick sono ben adeguati per il debugging, l'implementazione di semplici multitasking, backgrounded I/O e molti altri compiti.

Vedere anche register_tick_function() e unregister_tick_function().

return

Se viene chiamato all'interno di una funzione, l'istruzione return() termina immediatamente l'esecuzione della funzione corrente, e restituisce il suo argomento come valore della funzione chiamata. return() terminerà anche l'esecuzione di un'istruzione eval() o di un file di script.

Se viene chiamato in uno scope globale, allora verrà terrminata l'esecuzione del file di script corrente. Nel caso in cui il file di script corrente sia un file chiamato da include() o require(), il controllo viene passato al file chiamante. Ciononostante, se il file di script corrente è un file chiamato da include(), allora il valore dato da return() verrà restituito come valore della chiamata include(). Se viene chiamato return() all'interno del file di script principale, allora l'esecuzione dello script terminerà. Se il file di script corrente è stato nominato da auto_prepend_file o auto_append_file con le opzioni di configurazione nel file di configurazione, allora l'esecuzione di quello script termina.

Per maggiori informazioni, consultare Valori restituiti.

> **Nota:** Notate che poichè return() è un costrutto di linguaggio e non una funzione, le parentesi che circondano i suoi argomenti *non* sono richieste --infatti, è più comune evitarle che usarle, nonostante ciò non c'è motivo di preferire un modo o l'altro.

require()

L'istruzione require() include e valuta il file specifico.

require() include e valuta uno specifico file. Informazioni dettagliate su come funziona quest'inclusione sono descritte nella documentazione di include().

require() e include() sono identiche in ogni senso eccetto per come esse trattano gli errori. include() produce un Warning mentre require() restituisce un Fatal Error. In altre parole, non esitate ad usare require() se volete che un file mancante fermi l'esecuzione della pagina. include() non si comporta in questo modo, lo script continuerà nonostante tutto. Assicuratevi di avere un appropriato include_path impostato a dovere.

Esempio 11-2. Esempio di base con require()

```
<?php

require 'prepend.php';

require $qualche_file;

require ('qualche_file.txt');

?>
```

Vedere la documentazione di include() per più esempi.

> **Nota:** Prima di PHP 4.0.2, si applica la seguente logica: require() tenterà sempre di leggere il file chiamato, anche se la riga su cui si trova non verrà mai eseguita. L'istruzione condizionale non avrà effetto su require(). Comunque, se la riga su cui si verifica require() non viene eseguita, non sarà eseguito nemmeno il codice del file incluso. Similmente, le strutture cicliche non avranno effetto sul comportamento di require(). Sebbene il codice contenuto nel file incluso è ancora soggetto a ciclo, require() stesso si verifica solo una volta.

Vedere anche include(), require_once(), include_once(), eval(), file(), readfile(), virtual() e include_path.

include()

L'istruzione include() include e valuta il file specificato.

La documentazione seguente si applica anche a require(). I due costrutti sono identici in ogni aspetto eccetto per come essi trattano gli errori. include() produce un Warning mentre require() restituisce un Fatal Error. In altre parole, usate require() se volete che un file mancante fermi l'esecuzione della pagina. include() non si comporta in questo modo, lo script continuerà nonostante tutto. Assicuratevi di avere un appropriato include_path impostato a dovere.

Quando un file viene incluso, il codice che esso contiene eredita lo scope delle variabili della riga in cui si verifica l'inclusione. Qualsiasi variabile disponibile in quella riga nella chiamata al file sarà disponibile all'interno del file chiamato, da quel punto in avanti.

Esempio 11-3. Esempio di base con include()

```
vars.php
<?php

$colore = 'verde';
$frutto = 'mela';

?>

test.php
<?php

echo "Una $frutto $colore"; // Una

include 'vars.php';

echo "Una $frutto $colore"; // Una mela verde

?>
```

Se l'inclusione si verifica dentro una funzione all'interno del file chiamato, allora tutto il codice contenuto nel file chiamato si comporterà come se esso sia stato definito all'interno di una funzione. Così, esso seguirà lo scope delle variabili di quella funzione.

Esempio 11-4. Inclusione all'interno di funzioni

```
<?php

function foo()
{
global $frutto;

    include 'vars.php';

    echo "Una $frutto $colore";
}

/* vars.php è nello scope di foo() così    *
 * $colore NON è disponibile fuori di      *
 * questo scope. $frutto si perchè è stato *
 * dichiarato globale.                     */

foo();                        // Una mela verde
echo "Una $frutto $colore";   // Una mela

?>
```

Quando un file viene incluso, il parsing esce dalla modalità PHP e entra in modalità HTML all'inizio del file incluso, e riprende alla fine. Per questa ragione, qualunque codice all'interno del file incluso che dovrebbe essere eseguito come codice PHP deve essere incluso all'interno dei tag PHP validi di apertura e chiusura.

Se "URL fopen wrappers" nel PHP sono abilitati (come nella configurazione di default), potete specificare il file da includere usando un URL (via HTTP) invece che un percorso locale. Se il server chiamato interpreta il file incluso come codice PHP, le variabili possono essere passate al file incluso usando una stringa di richiesta URL come con l'utilizzo di HTTP GET. Non è proprio parlare della stessa cosa includere il file e averlo ereditato dallo scope di variabili del file chiamante; lo script è stato attualmente eseguito su un server remoto e il risultato è poi stato incluso nello script locale.

Esempio 11-5. include() attraverso HTTP

```
<?php

/* Questo esempio assume che www.example.com è configurato per eseguire file *
 * .php e non file .txt. Anche, 'Funziona' qui significa che le variabili    *
 * $foo e $bar sono disponibili all'interno del file incluso.                */

// Non funzionerà; file.txt non è stato trattato da www.example.com come PHP
include 'http://www.example.com/file.txt?foo=1&bar=2';

// Non funzionerà; cercare un file chiamato'file.php?foo=1&bar=2' nel
// filesystem locale.
include 'file.php?foo=1&bar=2';

// Funziona.
include 'http://www.example.com/file.php?foo=1&bar=2';

$foo = 1;
$bar = 2;
include 'file.txt';  // Funziona.
include 'file.php';  // Funziona.

?>
```

Vedere anche Remote files, fopen() e file() per informazioni correlate.

Poichè include() e require() sono speciali costrutti di linguaggio, dovete includerli all'interno di blocchi di istruzioni se si trovano in un blocco condizionale.

Esempio 11-6. include() e i blocchi condizionali

```
<?php

// Questo NON VA BENE e non funzionerà come desiderato.
if ($condizione)
    include $file;
else
    include $un_altro;
```

```
// Questo è CORRETTO.
if ($condizione) {
    include $file;
} else {
    include $un_altro;
}

?>
```

Trattamento dei valori restituiti: È possibile eseguire un'istruzione return() in un file incluso per terminare l'esecuzione di quel file e restituirlo allo script che l'ha chiamato. È anche possibile restituire valori dai file inclusi. Potete prendere il valore di una chiamata di inclusione come fareste con una normale funzione.

> **Nota:** In PHP 3, return potrebbe non apparire in un blocco a meno che esso sia un blocco di funzione, nel qual caso return() si applica a quella funzione e non all'intero file.

Esempio 11-7. include() and the return() statement

```
return.php
<?php

$var = 'PHP';

return $var;

?>

noreturn.php
<?php

$var = 'PHP';

?>

testreturns.php
<?php

$foo = include 'return.php';

echo $foo; // stampa 'PHP'

$bar = include 'noreturn.php';

echo $bar; // stampa 1

?>
```

`$bar` ha valore `1` perchè l'inclusione è stata eseguita con successo. Notare la differenza tra gli esempi sopra. Il primo usa return() all'interno di un file incluso mentre l'altro no. Pochi altri modi di "includere" file in variabili sono con fopen(), file() o usando include() insieme con Output Control Functions.

Vedere anche require(), require_once(), include_once(), readfile(), virtual(), e include_path.

require_once()

L'istruzione require_once() include e valuta il file specificato durante l'esecuzione dello script. È un comportamento simile all'istruzione require(), con la sola differenza che se il codice di un file è stato già incluso, esso non sarà incluso nuovamente. Vedere la documentazione di require() per maggiori informazioni su come funziona quest'istruzione.

require_once() dovrebbe essere usato nei casi dove lo stesso file potrebbe essere incluso e valutato più di una volta durante una particolare esecuzione di uno script, e volete essere sicuri che esso sia incluso esattamente una volta per evitare problemi con la ridefinizione di funzioni, riassegnazione di valori a variabili, etc.

Per esempi sull'utilizzo di require_once() e include_once(), consultare il codice PEAR (http://pear.php.net/) incluso nell'ultima distribuzione del codice sorgente di PHP.

> **Nota:** require_once() è stato aggiunto in PHP 4.0.1pl2

Vedere anche: require(), include(), include_once(), get_required_files(), get_included_files(), readfile(), e virtual().

include_once()

L'istruzione include_once() include e valuta il file specificato durante l'esecuzione dello script. È un comportamento simile all'istruzione include(), con la sola differenza che se il codice di un file è stato già incluso, esso non sarà incluso nuovamente. Come suggerisce il nome, esso sarà incluso solo una volta.

include_once() dovrebbe essere usato nei casi dove lo stesso file potrebbe essere incluso e valutato più di una volta durante una particolare esecuzione di uno script, e volete essere sicuri che esso sia incluso esattamente una volta per evitare problemi con la ridefinizione di funzioni, riassegnazione di valori a variabili, etc.

Per maggiori esempi sull'utilizzo di require_once() e include_once(), consultare il codice PEAR (http://pear.php.net/) incluso nell'ultima distribuzione del codice sorgente di PHP.

> **Nota:** include_once() è stato aggiunto in PHP 4.0.1pl2

Vedere anche include(), require(), require_once(), get_required_files(), get_included_files(), readfile(), e virtual().

Capitolo 12. Functions

User-defined functions

A function may be defined using syntax such as the following:

```
function foo ($arg_1, $arg_2, ..., $arg_n)
{
    echo "Example function.\n";
    return $retval;
}
```

Any valid PHP code may appear inside a function, even other functions and class definitions.

In PHP 3, functions must be defined before they are referenced. No such requirement exists in PHP 4.

PHP does not support function overloading, nor is it possible to undefine or redefine previously-declared functions.

PHP 3 does not support variable numbers of arguments to functions, although default arguments are supported (see Default argument values for more information). PHP 4 supports both: see Variable-length argument lists and the function references for func_num_args(), func_get_arg(), and func_get_args() for more information.

Function arguments

Information may be passed to functions via the argument list, which is a comma-delimited list of variables and/or constants.

PHP supports passing arguments by value (the default), passing by reference, and default argument values. Variable-length argument lists are supported only in PHP 4 and later; see Variable-length argument lists and the function references for func_num_args(), func_get_arg(), and func_get_args() for more information. A similar effect can be achieved in PHP 3 by passing an array of arguments to a function:

```
function takes_array($input)
{
    echo "$input[0] + $input[1] = ", $input[0]+$input[1];
}
```

Making arguments be passed by reference

By default, function arguments are passed by value (so that if you change the value of the argument within the function, it does not get changed outside of the function). If you wish to allow a function to modify its arguments, you must pass them by reference.

If you want an argument to a function to always be passed by reference, you can prepend an ampersand (&) to the argument name in the function definition:

```
function add_some_extra(&$string)
{
    $string .= 'and something extra.';
}
$str = 'This is a string, ';
add_some_extra($str);
echo $str;    // outputs 'This is a string, and something extra.'
```

Default argument values

A function may define C++-style default values for scalar arguments as follows:

```
function makecoffee ($type = "cappucino")
{
    return "Making a cup of $type.\n";
}
echo makecoffee ();
echo makecoffee ("espresso");
```

The output from the above snippet is:

```
Making a cup of cappucino.
Making a cup of espresso.
```

The default value must be a constant expression, not (for example) a variable or class member.

Note that when using default arguments, any defaults should be on the right side of any non-default arguments; otherwise, things will not work as expected. Consider the following code snippet:

```
function makeyogurt ($type = "acidophilus", $flavour)
{
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // won't work as expected
```

The output of the above example is:

```
Warning: Missing argument 2 in call to makeyogurt() in
/usr/local/etc/httpd/htdocs/php3test/functest.html on line 41
Making a bowl of raspberry .
```

Now, compare the above with this:

```
function makeyogurt ($flavour, $type = "acidophilus")
{
    return "Making a bowl of $type $flavour.\n";
}

echo makeyogurt ("raspberry");   // works as expected
```

The output of this example is:

```
Making a bowl of acidophilus raspberry.
```

Variable-length argument lists

PHP 4 has support for variable-length argument lists in user-defined functions. This is really quite easy, using the func_num_args(), func_get_arg(), and func_get_args() functions.

No special syntax is required, and argument lists may still be explicitly provided with function definitions and will behave as normal.

Returning values

Values are returned by using the optional return statement. Any type may be returned, including lists and objects. This causes the function to end its execution immediately and pass control back to the line from which it was called. See return() for more information.

```
function square ($num)
{
    return $num * $num;
}
echo square (4);   // outputs ’16’.
```

You can't return multiple values from a function, but similar results can be obtained by returning a list.

```
function small_numbers()
{
    return array (0, 1, 2);
}
list ($zero, $one, $two) = small_numbers();
```

To return a reference from a function, you have to use the reference operator & in both the function declaration and when assigning the returned value to a variable:

```
function &returns_reference()
{
    return $someref;
}

$newref =& returns_reference();
```

For more information on references, please check out References Explained.

old_function

The `old_function` statement allows you to declare a function using a syntax identical to PHP/FI2 (except you must replace 'function' with 'old_function'.

This is a deprecated feature, and should only be used by the PHP/FI2->PHP 3 convertor.

> **Attenzione**
>
> Functions declared as `old_function` cannot be called from PHP's internal code. Among other things, this means you can't use them in functions such as usort(), array_walk(), and register_shutdown_function(). You can get around this limitation by writing a wrapper function (in normal PHP 3 form) to call the `old_function`.

Variable functions

PHP supports the concept of variable functions. This means that if a variable name has parentheses appended to it, PHP will look for a function with the same name as whatever the variable evaluates

to, and will attempt to execute it. Among other things, this can be used to implement callbacks, function tables, and so forth.

Variable functions won't work with language constructs such as echo(), unset(), isset(), empty() and include(). Although, the construct print() is an exception and will work. This is one of the major differences between PHP functions and language constructs.

Esempio 12-1. Variable function example

```
<?php
function foo()
{
    echo "In foo()<br>\n";
}

function bar($arg = ”)
{
    echo "In bar(); argument was ’$arg’.<br>\n";
}

$func = ’foo’;
$func();
$func = ’bar’;
$func(’test’);
?>
```

See also variable variables and function_exists().

Capitolo 13. Classi e Oggetti

Classi

Una classe è una collezione di variabili e funzioni che utilizzano queste variabili. Una classe si definisce usando la seguente sintassi:

```
<?php
class Cart
{
    var $items;  // Articoli nel carrello

    // Aggiunge $num articoli di $artnr nel carrello

    function add_item ($artnr, $num)
    {
        $this->items[$artnr] += $num;
    }

    // Prende $num articoli di $artnr e li rimuove dal carrello

    function remove_item ($artnr, $num)
    {
        if ($this->items[$artnr] > $num) {
            $this->items[$artnr] -= $num;
            return true;
        } else {
            return false;
        }
    }
}
?>
```

Il codice definisce una classe chiamata Cart composta da un array associativo che archivia gli articoli nel carrello e due funzioni per aggiungere e rimuovere gli articoli dal carrello stesso.

> **Cautela**
>
> Le seguenti note cautelari sono valide per PHP 4.
>
> Il nome `stdClass` è usato esclusivamente da Zend ed è riservato. Non è quindi possibile creare una classe chiamata `stdClass` in PHP.
>
> I nomi di funzione `__sleep` e `__wakeup` sono riservati e magici nelle classi PHP. Non è possibile creare funzioni con questi nomi nelle classi definite dall'utente, a meno che non sia desiderata la funzionalità magica connessa a questi nomi. Si veda sotto per avere più informazioni.
>
> PHP riserva tutti i nomi di funzione che iniziano con __ a funzioni magiche. Si suggerisce di non usare nomi di funzioni che utilizzano con i caratteri __ in PHP a meno che non si desideri implementare una funzionalità magica.

Nota: In PHP 4, sono permesse inizializzazioni di variabili con valori costanti solamente grazie all'uso di `var`. Per inizializzare variabili con valori non-costanti, bisogna creare una funzione

d’inizializzazione che è chiamata automaticamente all’istanziazione di un oggetto da una classe. Questo tipo di funzione si chiama costruttore (vedi sotto).

```
<?php
/* questo non funziona in PHP 4. */
class Cart
{
    var $todays_date = date("Y-m-d");
    var $name = $firstname;
    var $owner = ’Fred ’ . ’Jones’;
    var $items = array("VCR", "TV");
}

/* Questo è corretto. */
class Cart
{
    var $todays_date;
    var $name;
    var $owner;
    var $items;

    function Cart()
    {
        $this->todays_date = date("Y-m-d");
        $this->name = $GLOBALS[’firstname’];
        /* etc ... */
    }
}
```

Le classi sono tipi del linguaggio, e sono modelli per variabili reali. Per creare una variabile oggetto si usa l’operatore `new`.

```
<?php
$cart = new Cart;
$cart->add_item("10", 1);

$another_cart = new Cart;
$another_cart->add_item("0815", 3);
```

Il codice sopra, genera gli oggetti $cart e $another_cart, dalla classe Cart. La funzione add_item() dell’oggetto $cart è chiamata per aggiungere una ricorrenza dell’articolo numero 10 a $cart. Ad $another_cart sono aggiunte 3 ricorrenze dell’articolo numero 0815.

Sia $cart che $another_cart dispongono delle funzioni add_item(), remove_item() e della variabile $items, ma per ogni oggetto queste sono funzioni e variabili sono distinte. Potete pensare agli oggetti come a qualcosa di simile alle directories di un filesystem. In un filesystem si possono avere due diversi files README.TXT, purchè siano in directories differenti. Così come in un filesystem dovete digitare il nome (percorso) completo per raggiungere un determinato file partendo da una directory toplevel, così dovete specificare il nome completo di una funzione o variabile che desiderate richiamare da un oggetto. Per PHP, la directory toplevel è il namespace globale dell’oggetto ed il separatore del pathname (/) è ->. Così $cart->items e $another_cart->items sono due diverse variabili

che differiscono per il nome. Si noti che la variabile si chiama $cart->items, e non $cart->$items, questo perchè le variabili il PHP si scrivono con un unico simbolo di dollaro.

```
// corretto, con un simbolo $
$cart->items = array("10" => 1);

// non valido, perchè $cart->$items diventa $cart->""
$cart->$items = array("10" => 1);

// corretto, ma non sempre può funzionare:
// $cart->$myvar diventa $cart->items
$myvar = ’items’;
$cart->$myvar = array("10" => 1);
```

Quando si definisce una classe, non è possibile prevedere quale nome avrà l’oggetto istanziato nel programma. Quando la classe Cart è stata scritta, non si poteva prevedere che l’oggetto istanziato da essa si sarebbe potuto chiamare $cart o $another_cart. Quindi non è possibile scrivere $cart->items all’interno della classe Cart in fase di progettazione. Per poter accedere alle funzioni e alle variabili interne di una classe perciò si usa la pseudo-variabile $this che può essere letta come ’la mia\il mio’ o ’di questo oggetto’. Quindi, ’$this->items[$artnr] += $num’ può essere letto come ’aggiungi $num al contatore $artnr al del mio array degli articoli’ o ’aggiungi $num al contatore $artnr dell’array degli articoli di questo oggetto’.

> **Nota:** Ci sono molte utili funzioni per manipolare classi ed oggetti. Se desiderate conoscerle potete dare un’occhiata alle Class/Object Functions

extends

Spesso si ha bisogno di avere classi con variabili e funzioni simili ad altre classi. É buona norma definire una classe in modo generico, sia per poterla riutilizzare spesso, sia per poterla adattare a scopi specifici.Per facilitare questa operazione, è possibile generare classi per estensione di altre classi. Una classe estesa o derivata ha tutte le variabili e le funzioni della classe di base (questo fenomeno è chiamato ’eredità’, anche se non muore nessuno) più tutto ciò che viene aggiunto dall’estensione. Non è possibile che una sottoclasse, ridefinisca variabili e funzioni di una classe madre. Una classe estesa dipende sempre da una singola classe di base: l’eredità multipla non è supportata. Le classi si estendono usando la parola chiave ’extends’.

```
class Named_Cart extends Cart
{
    var $owner;

    function set_owner ($name)
    {
        $this->owner = $name;
    }
}
```

Qui viene definita una classe Named_Cart che ha tutte le funzioni e variabili di Cart più la variabile $owner e la funzione set_owner(). Viene creato un carrello con nome con il metodo usato in precedenza, in più la classe estesa permette di settare o leggere il nome del carrello. Si possono usare variabili e funzioni sia di Cart che della sua estensione:

```
$ncart = new Named_Cart;    // Crea un carrello con nome
$ncart->set_owner("kris");  // Assegna il nome al carrello
print $ncart->owner;        // stampa il nome del proprietario
$ncart->add_item("10", 1);  // (funzionalità ereditata da Cart)
```

Costruttori

> **Cautela**
>
> In PHP 3 i costruttori si comportano diversamente rispetto a PHP 4. La semantica di PHP 4 è decisamente da preferire.

I costruttori sono funzioni che esistono in una classe e che sono chiamate automaticamente quando si crea una nuova istanza di una classe con new. In PHP 3, una funzione si transforma in in un costruttore quando ha lo stesso nome di una classe. In PHP 4, una funzione diventa un costruttore, quando ha lo stesso nome di una classe ed è definita all'interno della classe stessa - la differenza è sottile, ma cruciale (si veda sotto).

```
// Funziona in PHP 3 e PHP 4.
class Auto_Cart extends Cart
{
    function Auto_Cart()
    {
        $this->add_item ("10", 1);
    }
}
```

Questo codice definisce una classe Auto_Cart, che non è altro che Cart più un costruttore che inizializza il carrello con una occorrenza dell'articolo numero "10" ogni volta che un nuovo Auto_Cart è creato con "new". I costruttori possono avere degli argomenti, e gli argomenti possono essere facoltativi, questo li rende molto versatili. Per poter usare una classe senza specificare parametri, tutti i parametri del costruttore devono essere resi facoltativi con valori di default.

```
// Funziona in PHP 3 e PHP 4.
class Constructor_Cart extends Cart
{
    function Constructor_Cart($item = "10", $num = 1)
    {
```

```
        $this->add_item ($item, $num);
    }
}

// Istanzia il vecchio e noioso carrello.

$default_cart = new Constructor_Cart;

// Un carrello nuovo ...

$different_cart = new Constructor_Cart("20", 17);
```

Cautela

In PHP 3, le classi e i costruttori derivati presentano un certo numero di limitazioni. I seguenti esempi dovrebbero essere letti con attenzione per capire queste limitazioni.

```
class A
{
    function A()
    {
      echo "Sono il costtruttore di A.<br>\n";
    }
}

class B extends A
{
    function C()
    {
        echo "Sono una normale funzione.<br>\n";
    }
}

// nessun costruttore è chiamato in PHP 3.
$b = new B;
```

In PHP 3, nessun costruttore è stato chiamanto nel suddetto esempio. La regola in PHP 3 è: 'un costruttore è una funzione che ha lo stesso nome di una classe'. Il nome della classe è B e non c'è nessuna funzione B() nella classe B.

Questa regola è stata cambiata in PHP 4, la nuova regola dice: 'Se una classe non ha un costruttore proprio, verrà chiamato il costruttore della classe base, se esiste'. Il suddetto esempio avrebbe stampato 'Sono il costruttore di A.' in PHP 4.

```
class A
{
    function A()
    {
```

```
        echo "Sono il costruttore di A.<br>\n";
    }

    function B()
    {
        echo "Sono una normale funzione di nome B della classe A.<br>\n";
        echo "Non sono il costruttore di A.<br>\n";
    }
}

class B extends A
{
    function C()
    {
        echo "Sono una normale funzione.<br>\n";
    }
}

// This will call B() as a constructor.
$b = new B;
```

In PHP 3, la funzione B() della classe A si transformerà improvvisamente in un costruttore per la classe B, anche se questo non era previsto. La regola in PHP 3 è: ’un costruttore è una funzione che ha lo stesso nome di una classe’. PHP 3 non si preoccupa se la funzione è stata definita nella classe B o se è stata ereditata.

La regola è stata corretta in PHP 4 ed è diventata: ’un costruttore è una funzione con lo stesso nome della classe in cui è definita’. Così in PHP 4, la classe B non avendo una funzione costruttore avrebbe richiamato il costruttore della sua classe base, e sarebbe stato stampato ’io sono il costruttore di A.’.

> **Cautela**
>
> Nè PHP 3 nè PHP 4 chiamano costruttori di una classe base automaticamente da un costruttore di una classe derivata. È responsabilità del programmatore propagare la chiamata ai costruttori dove è necessario.

Nota: Non ci sono distruttori in PHP 3 o in PHP 4. Potete usare register_shutdown_function() per simulare la maggior parte degli effetti dei distruttori.

I distruttori sono funzioni che sono chiamate automaticamente quando una variabile è distrutta con unset() o semplicemente uscendo dal suo ambito. Non ci sono distruttori in PHP.

::

> **Cautela**
>
> Ciò che segue è valido soltanto per PHP 4.

A volte è utile riferirsi alle funzioni ed alle variabili di classi base o riferirsi alle funzioni di classi senza istanziarle. L'operatore :: è usato per questi scopi.

```
class A
{
    function example()
    {
        echo "Sono la funzione originale A::example().<br>\n";
    }
}

class B extends A
{
    function example()
    {
        echo "Sono la funzione ridefinita B::example().<br>\n";
        A::example();
    }
}

// non viene istanziato nessun oggetto dalla classe A.
// ma il codice stampa
//   Sono la funzione originale A::example().<br>
A::example();

// crea un oggetto dalla classe B.
$b = new B;

// questo codice stampa
//   Sono la funzione ridefinita B::example().<br>
//   Sono la funzione originale A::example().<br>
$b->example();
```

L'esempio chiama la funzione example() della classe A, ma senza creare un'istanza di A, di modo che la funzione non si possa richiamare con $a->example(). example() è chiamata come 'funzione della classe', e non come funzione di un oggetto della classe.

Si possono usare funzioni della classe, ma non le variabili della classe. Infatti, non esiste nessun oggetto nel momento della chiamata della funzione. Quindi, la funzione della classe non può usare le variabili dell'oggetto (ma può usare le variabili locali e globali) e $this non può essere usato.

Nel suddetto esempio, la classe B ridefinisce la funzione example(). La funzione originale definita nella classe A è adombrata e non più disponibile, a meno che voi non chiamiate esplicitamente con l'operatore :: scrivendo A::example() per richiamare la funzione (è possibile anche scrivere parent::example(), come mostra la sezione seguente).

In questo contesto, c'è un oggetto corrente che può avere determinate variabili. Una volta usate da parte di una funzione dell'oggetto, potete usare $this per le variabili dell'oggetto.

`parent`

E possibile ritrovarsi a scrivere classi con codice che si riferisce a variabili e funzioni di classi base. Ciò è particolarmente VERO se una classe derivata è un perfezionamento o una specializzazione di una classe base.

Invece di usare il nome letterale della classe, bisognerebbe usare il nome speciale `parent`, che si riferisce al nome della classe base definita nella dichiarazione di `extends`. Usando questo metodo, si evita di usare il nome della classe base nel codice scritto. Se l'albero di eredità cambiasse durante lo sviluppo della classe, il cambiamento si ridurrebbe semplicemente alla modifica della dichiarazione `extends` della classe.

```
class A
{
    function example()
    {
        echo "Sono A::example() e fornisco una funzionalità di base.<br>\n";
    }
}

class B extends A
{
    function example()
    {
        echo "Sono B::example() e fornisco una funzionalità aggiuntiva.<br>\n";
        parent::example();
    }
}

$b = new B;

// Il codice chiama B::example(), che a sua volta chiama A::example().
$b->example();
```

Serializzare oggetti - oggetti nelle sessioni

Nota: In PHP 3, gli oggetti perdono la loro associazione di classe durante il processo di serializzazione e di deserializzazione. La variabile risultante è di tipo oggetto, ma non ha classe e metodi, e diventa inutile (come un buffo array).

> **Cautela**
>
> Le seguenti informazioni sono valide soltanto per PHP 4.

serialize() restituisce una stringa che contiene una rappresentazione byte-stream di tutti i valori che possono essere memorizzati in PHP. unserialize() può usare questa stringa per ricreare i valori variabili utilizzabili. Usando serialize() per salvare un oggetto si salveranno tutte le variabili dell'oggetto. Le funzioni dell'oggetto non sono salvate, viene salvato solo il nome della classe.

Per potere usare unserialize() su un oggetto, la classe dell'oggetto deve essere definita. Cioè se avete un oggetto $a della classe A su una pagina di nome page1.php e usate serialize(), otterrete una stringa che si riferisce alla classe A e contiene tutti i valori delle variabili contenute in $a. Se desiderate potere deserializzare l'oggetto in un'altra pagina chiamata page2.php, dovete ricreare $a dalla classe A, la definizione della classe A perciò deve essere presente nella pagina page2.php. Questo può essere fatto per esempio memorizzando la definizione della classe A in un file che viene incluso sia in page1.php che in page2.php.

```
classa.inc:
  class A
  {
      var $one = 1;

      function show_one()
      {
          echo $this->one;
      }
  }

page1.php:
  include("classa.inc");

  $a = new A;
  $s = serialize($a);
  // memorizzare $s in qualche luogo della page2.
  $fp = fopen("store", "w");
  fputs($fp, $s);
  fclose($fp);

page2.php:
  // questo è necessario perchè unserialize() funzioni correttamente.
  include("classa.inc");

  $s = implode("", @file("store"));
  $a = unserialize($s);

  // ora usiamo la function show_one() dell'oggetto $a.
  $a->show_one();
```

Se state usando le sessioni ed usate session_register() per registrare oggetti, questi oggetti vengono serializzati automaticamente alla fine di ogni pagina PHP e sono deserializzate automaticamente su ogni pagina della sessione. Ciò significa che gli oggetti possono mostrarsi in ogni pagina e che sono parte integrante della sessione.

Si suggerisce vivamente di includere le definizioni delle classi degli oggetti registrati su tutte le pagine, anche se le classi non sono usate su tutte le pagine. Se un oggetto viene deserializzato senza la relativa definizione della classe, perderà l'associazione ad essa e si transformerà in in un oggetto della classe `stdClass` senza nessuna funzione disponibile, diventando inutile.

Così se nell'esempio qui sopra $a diventasse parte di una sessione e fosse registrato con `session_register("a")`, dovreste includere un file `classa.inc` su tutte le pagine in cui è valida la sessione, non soltanto nella page1.php e nella page2.php.

Le funzioni magiche `__sleep` e `__wakeup`

serialize() controlla se la vostra classe ha una funzione dal nome magico `__sleep`. In caso affermativo, quella funzione viene eseguita prima di qualsiasi serializzazione. La funzione può pulire l'oggetto e restituire un array con i nomi di tutte le variabili di quell' oggetto che dovrebbero essere serializzate.

Si intende usare `__sleep` quando chiudendo un collegamento ad un database l'oggetto può avere dati pendenti e l'oggetto ha bisogno di essere ripulito. Inoltre, la funzione è utile se avete oggetti molto grandi che non devono essere salvati completamente.

Per contro, unserialize() controlla per vedere se c'è nella classe una funzione dal nome magico `__wakeup`. Se è presente questa funzione può ricostruire qualunque risorsa che l'oggetto aveva.

L'intento di `__wakeup` è quello di ristabilire le connessioni ai database che possono esser state persi durante la serializzazione ed effettuare altre mansioni reinizializzazione.

Riferimenti all'interno del costruttore

La creazione di riferimenti con costruttori può condurre a risultati confusi. Questa sezione in stile Tutorial vi aiuterà ad evitare problemi.

```
class Foo
{
    function Foo($name)
    {
        // crea un riferimento all'interno della variabile $globalref
        global $globalref;
        $globalref[] = &$this;
        // setta Name con il valore passato
        $this->setName($name);
        // e lo manda all'output
        $this->echoName();
    }

    function echoName()
    {
        echo "<br>",$this->name;
    }

    function setName($name)
    {
        $this->name = $name;
    }
}
```

Verifichiamo se c'è una differenza fra `$bar1` che è stato creato usando l'operatore `=` e `$bar2` che è stato creato usando l'operatore di riferimento `=&` ...

```
$bar1 = new Foo('set in constructor');
$bar1->echoName();
$globalref[0]->echoName();

/* output:
imposta nel costruttore
imposta nel costruttore
imposta nel costruttore */

$bar2 =& new Foo('set in constructor');
$bar2->echoName();
$globalref[1]->echoName();

/* output:
imposta nel costruttore
imposta nel costruttore
imposta nel costruttore */
```

Apparentemente non c'è differenza, ed in effetti questo è molto significativo: `$bar1` e `$globalref[0]` _ NON _ sono riferimenti, ma sono due variabili diverse. Questo succede perché "new" non restituisce per default un riferimento, ma restituisce una copia.

> **Nota:** Non c'è perdita di prestazioni (da php 4 in su si usa il riferimento) ad istanziare copie per riferimento. Al contrario spesso è meglio lavorare con copie istanziate per riferimento, perché creare copie reali richiede un certo tempo, mentre creare riferimenti virtuali è immediato, (a meno che non si parli di un grande array o un oggetto che viene modificato in modo successivo, allora sarebbe saggio usare i riferimenti per cambiargli tutti i valori simultaneamente).

Per dimostrare quello che è scritto sopra guardate il codice qui sotto.

```
// ora cambieremo il nome che cosa vi aspettate?
// potreste prevedere che $bar e $globalref[0] cambino i loro nomi ...
$bar1->setName('set from outside');

// come accennato prima ecco il risultato.
$bar1->echoName();
$globalref[0]->echoName();

/* output:
set from outside
set in constructor */

// vediamo le differenze tra $bar2 e $globalref[1]
$bar2->setName('set from outside');

// fortunatamen sono solo uguali, ma sono la stessa variabile
// $bar2->name e $globalref[1]->name sono la stessa cosa
$bar2->echoName();
$globalref[1]->echoName();

/* output:
set from outside
```

```
set from outside */
```

Un esempio finale, prova a farvi capire.

```
class A
{
    function A($i)
    {
        $this->value = $i;
        // provare a capire perchè qui non abbiamo bisogno d'un riferimento
        $this->b = new B($this);
    }

    function createRef()
    {
        $this->c = new B($this);
    }

    function echoValue()
    {
        echo "<br>","class ",get_class($this),': ',$this->value;
    }
}


class B
{
    function B(&$a)
    {
        $this->a = &$a;
    }

    function echoValue()
    {
        echo "<br>","class ",get_class($this),': ',$this->a->value;
    }
}

// prova a capire perchè usando una semplice copia si avrebbe
// in un risultato indesiderato nella riga segnata con *
$a =& new A(10);
$a->createRef();

$a->echoValue();
$a->b->echoValue();
$a->c->echoValue();

$a->value = 11;

$a->echoValue();
$a->b->echoValue(); // *
$a->c->echoValue();
```

```
/*
output:
class A: 10
class B: 10
class B: 10
class A: 11
class B: 11
class B: 11
*/
```

Capitolo 14. References Explained

What References Are

References are a means in PHP to access the same variable content by different names. They are not like C pointers, they are symbol table aliases. Note that in PHP, variable name and variable content are different, so the same content can have different names. The most close analogy is with Unix filenames and files - variable names are directory entries, while variable contents is the file itself. References can be thought of as hardlinking in Unix filesystem.

What References Do

PHP references allow you to make two variables to refer to the same content. Meaning, when you do:

```
$a =& $b
```

it means that `$a` and `$b` point to the same variable.

> **Nota:** `$a` and `$b` are completely equal here, that's not `$a` is pointing to `$b` or vice versa, that's `$a` and `$b` pointing to the same place.

The same syntax can be used with functions, that return references, and with `new` operator (in PHP 4.0.4 and later):

```
$bar =& new fooclass();
$foo =& find_var ($bar);
```

> **Nota:** Not using the `&` operator causes a copy of the object to be made. If you use `$this` in the class it will operate on the current instance of the class. The assignment without `&` will copy the instance (i.e. the object) and `$this` will operate on the copy, which is not always what is desired. Usually you want to have a single instance to work with, due to performance and memory consumption issues.

The second thing references do is to pass variables by-reference. This is done by making a local variable in a function and a variable in the calling scope reference to the same content. Example:

```
function foo (&$var)
{
    $var++;
```

```
}

$a=5;
foo ($a);
```

will make `$a` to be 6. This happens because in the function `foo` the variable `$var` refers to the same content as `$a`. See also more detailed explanations about passing by reference.

The third thing reference can do is return by reference.

What References Are Not

As said before, references aren't pointers. That means, the following construct won't do what you expect:

```
function foo (&$var)
{
    $var =& $GLOBALS["baz"];
}
foo($bar);
```

What happens is that `$var` in `foo` will be bound with `$bar` in caller, but then it will be re-bound with `$GLOBALS["baz"]`. There's no way to bind `$bar` in the calling scope to something else using the reference mechanism, since `$bar` is not available in the function `foo` (it is represented by `$var`, but `$var` has only variable contents and not name-to-value binding in the calling symbol table).

Passing by Reference

You can pass variable to function by reference, so that function could modify its arguments. The syntax is as follows:

```
function foo (&$var)
{
    $var++;
}

$a=5;
foo ($a);
// $a is 6 here
```

Note that there's no reference sign on function call - only on function definition. Function definition alone is enough to correctly pass the argument by reference.

Following things can be passed by reference:

- Variable, i.e. `foo($a)`
- New statement, i.e. `foo(new foobar())`
- Reference, returned from a function, i.e.:

```
function &bar()
{
    $a = 5;
    return $a;
}
foo(bar());
```

  See also explanations about returning by reference.

Any other expression should not be passed by reference, as the result is undefined. For example, the following examples of passing by reference are invalid:

```
function bar() // Note the missing &
{
    $a = 5;
    return $a;
}
foo(bar());

foo($a = 5) // Expression, not variable
foo(5) // Constant, not variable
```

These requirements are for PHP 4.0.4 and later.

Returning References

Returning by-reference is useful when you want to use a function to find which variable a reference should be bound to. When returning references, use this syntax:

```
function &find_var ($param)
{
    ...code...
    return $found_var;
}

$foo =& find_var ($bar);
$foo->x = 2;
```

In this example, the property of the object returned by the `find_var` function would be set, not the copy, as it would be without using reference syntax.

> **Nota:** Unlike parameter passing, here you have to use `&` in both places - to indicate that you return by-reference, not a copy as usual, and to indicate that reference binding, rather than usual assignment, should be done for `$foo`.

Unsetting References

When you unset the reference, you just break the binding between variable name and variable content. This does not mean that variable content will be destroyed. For example:

```
$a = 1;
$b =& $a;
unset ($a);
```

won't unset `$b`, just `$a`.

Again, it might be useful to think about this as analogous to Unix **unlink** call.

Spotting References

Many syntax constructs in PHP are implemented via referencing mechanisms, so everything told above about reference binding also apply to these constructs. Some constructs, like passing and returning by-reference, are mentioned above. Other constructs that use references are:

`global` References

When you declare variable as **global $var** you are in fact creating reference to a global variable. That means, this is the same as:

```
$var =& $GLOBALS["var"];
```

That means, for example, that unsetting `$var` won't unset global variable.

`$this`

In an object method, `$this` is always reference to the caller object.

Parte III. Caratteristiche

Capitolo 15. Gestione degli errori

In PHP sono presenti diversi tipi di errori e avvertimenti (warning):

Tabella 15-1. PHP - Tipi di errore

valore	simbolo	descrizione	note
1	E_ERROR	Errore run-time fatale	
2	E_WARNING	Warning run-time(errore non fatale)	
4	E_PARSE	Errore nel parsing in compilazione	
8	E_NOTICE	Notifiche run-time (meno serie dei warning)	
16	E_CORE_ERROR	Errore fatale nella fase di startup inizale di PHP	Solo PHP 4
32	E_CORE_WARNING	Warning (errore non fatale) nella fase di startup inizale di PHP	Solo PHP 4
64	E_COMPILE_ERROR	Errore fatale in fase di compilazione	Solo PHP 4
128	E_COMPILE_WARNING	Warning (errore non fatale) in fase di compilazione	Solo PHP 4
256	E_USER_ERROR	Messagio di errore generato dall'utente	Solo PHP 4
512	E_USER_WARNING	Messaggio di avvertimento (warning) generato dall'utente	Solo PHP 4
1024	E_USER_NOTICE	Messagio di notifica generato dall'utente	Solo PHP 4
	E_ALL	Tutti i precedenti	Solo PHP 4

I valori presenti nella tabella (sia numerici che simbolici) sono utilizzati per creare delle bitmask per specificare quali errori segnalare. Si possono usare gli operatori sui bit '|','&' e '~' per combinare questi valori e mascherare certi tipi di errori. Nota che solo '|', '~', '!', e '&' verranno interpretati correttamente all'interno del file `php.ini` e che gli operatori sui bit non saranno interpretati nel file `php3.ini`.

Nel PHP 4 la configurazione predefinita di error_reporting è `E_ALL & ~E_NOTICE` che fa si che vengano visualizzati tutti gli errori e avvertimenti che non siano di livello E_NOTICE. Nel PHP 3 la configurazione predefinita è `(E_ERROR | E_WARNING | E_PARSE)`, che ha lo stesso effetto. Si noti che, dato che le costanti non sono supportate nel file `php3.ini` presente nel PHP 3, la configurazione di error_reporting va effettuata usando un valore numerico, ad esempio `7`.

Le configurazioni predefinite possono essere cambiate nel file ini con la direttiva error_reporting. Si può anche utilizzare il file di configurazione di Apache `httpd.conf` con la direttiva php_error_reporting (php3_error_reporting per PHP 3) oppure ancora in fase di esecuzione di uno script con la funzione error_reporting().

> **Attenzione**
>
> Quando si esegue un upgrade del codice o dei server da PHP3 a PHP4 è necessario controllare questi settaggi e le chiamate a error_reporting() oppure potrebbe disabilitarsi il report dei nuovi tipi di errore, specialmente E_COMPILE_ERROR. Questo potrebbe portare a documenti vuoti senza alcun feedback sulle cause o dove guardare per trovare il problema.

Tutte le espressioni PHP possono anche venir chiamate con il prefisso "@", che disabilita il report degli errori per quella particolare espressione. Se capita un errore in una di queste espressioni e l'opzione track_errors è attivata, si può trovare il messaggio d'errore nella variabile globale `$php_errormsg`.

> **Nota:** Il prefisso @ non disabilita i messaggi che sono il risultato di errori di parsing.

> **Attenzione**
>
> Attualmente il prefisso @ disabilita anche il report per gli errori critici che terminano l'esecuzione dello script. Fra le altre cose, questo significa che se si usa @ per sopprimere gli errori di una determinata funzione e questa è ad esempio non disponibile oppure è stata chiamata in maniera non corretta, lo script terminerà e non ci sarà nessuna indicazione del perché.

Di seguito si può vedere un esempio di come si possono usare le possibilità di gestione degli errori del PHP. Definiamo una funzione per la gestione degli errori che registra le informazioni in un file (utilizzando un formato XML) e invia una email allo sviluppatore quando si verifica un errore critico nella logica.

Esempio 15-1. Usare la gestione degli errori in uno script

```
<?php
// sviluppiamo la nostra gestione degli errori
error_reporting(0);

// funzione per la gestione degli errori definita dall'utente
function userErrorHandler ($errno, $errmsg, $filename, $linenum, $vars) {
    // data e ora della annotazione dell'errore
    $dt = date("Y-m-d H:i:s (T)");

    // define an assoc array of error string
    // in reality the only entries we should
    // consider are 2,8,256,512 and 1024
    $errortype = array (
                1   =>  "Error",
                2   =>  "Warning",
                4   =>  "Parsing Error",
                8   =>  "Notice",
                16  =>  "Core Error",
                32  =>  "Core Warning",
                64  =>  "Compile Error",
                128 =>  "Compile Warning",
```

```
                256 =>  "User Error",
                512 =>  "User Warning",
                1024=>  "User Notice"
                );
    // set of errors for which a var trace will be saved
    $user_errors = array(E_USER_ERROR, E_USER_WARNING, E_USER_NOTICE);

    $err = "<errorentry>\n";
    $err .= "\t<datetime>".$dt."</datetime>\n";
    $err .= "\t<errornum>".$errno."</errornum>\n";
    $err .= "\t<errortype>".$errortype[$errno]."</errortype>\n";
    $err .= "\t<errormsg>".$errmsg."</errormsg>\n";
    $err .= "\t<scriptname>".$filename."</scriptname>\n";
    $err .= "\t<scriptlinenum>".$linenum."</scriptlinenum>\n";

    if (in_array($errno, $user_errors))
        $err .= "\t<vartrace>".wddx_serialize_value($vars,"Variables")."</vartrace>\n";
    $err .= "</errorentry>\n\n";

    // for testing
    // echo $err;

    // save to the error log, and e-mail me if there is a critical user error
    error_log($err, 3, "/usr/local/php4/error.log");
    if ($errno == E_USER_ERROR)
        mail("phpdev@example.com","Critical User Error",$err);
}


function distance ($vect1, $vect2) {
    if (!is_array($vect1) || !is_array($vect2)) {
        trigger_error("Incorrect parameters, arrays expected", E_USER_ERROR);
        return NULL;
    }

    if (count($vect1) != count($vect2)) {
        trigger_error("Vectors need to be of the same size", E_USER_ERROR);
        return NULL;
    }

    for ($i=0; $i<count($vect1); $i++) {
        $c1 = $vect1[$i]; $c2 = $vect2[$i];
        $d = 0.0;
        if (!is_numeric($c1)) {
            trigger_error("Coordinate $i in vector 1 is not a number, using zero",
                            E_USER_WARNING);
            $c1 = 0.0;
        }
        if (!is_numeric($c2)) {
            trigger_error("Coordinate $i in vector 2 is not a number, using zero",
                            E_USER_WARNING);
            $c2 = 0.0;
        }
        $d += $c2*$c2 - $c1*$c1;
    }
    return sqrt($d);
}
```

```
$old_error_handler = set_error_handler("userErrorHandler");

// undefined constant, generates a warning
$t = I_AM_NOT_DEFINED;

// define some "vectors"
$a = array(2,3,"foo");
$b = array(5.5, 4.3, -1.6);
$c = array (1,-3);

// generate a user error
$t1 = distance($c,$b)."\n";

// generate another user error
$t2 = distance($b,"i am not an array")."\n";

// generate a warning
$t3 = distance($a,$b)."\n";

?>
```

Questo è un semplice esempio che mostra come usare le funzioni per la gestione degli errori e per loggare.

Vedere anche error_reporting(), error_log(), set_error_handler(), restore_error_handler(), trigger_error(), user_error()

Capitolo 16. Creazione e manipolazione di immagini

PHP non è limitato alla creazione di output HTML. Può anche essere usato per creare e manipolare file di immagini in una varietà di differenti formati, inclusi gif, png, jpg, wbmp e xpm. Ancora più convenientemente, php può visualizzare un'immagine da esso creata, direttamente in un browser. E' necessario compilare PHP con le librerie GD per poter usare queste funzioni. GD e PHP potrebbero necessitare di altre librerie, a seconda di quali formati immagine si desidera usare. GD ha smesso di supportare le immagini Gif dalla versione 1.6.

Esempio 16-1. Creazione di PNG usando PHP

```
<?php
    Header("Content-type: image/png");
    $stringa=implode($argv," ");
    $im = ImageCreateFromPng("immagini/bottone1.png");
    $arancio = ImageColorAllocate($im, 220, 210, 60);
    $px = (imagesx($im)-7.5*strlen($string))/2;
    ImageString($im,3,$px,9,$stringa,$arancio);
    ImagePng($im);
    ImageDestroy($im);
?>
```

Questo esempio può essere richiamato da una pagina con un tag tipo: <img src="bottone.php?testo"> Lo script bottone.php presentato sopra prende la stringa "testo" e la sovrappone ad una immagine base che, in questo caso, è "immagini/bottone1.png" e visualizza l'immagine risultante. Questo è un modo molto conveniente per evitare di disegnare nuove immagini di bottoni ogni volta che si desidera modificare il testo di un bottone. Con questo metodo esse vengono generate dinamicamente.

Capitolo 17. Autenticazione HTTP usando PHP

I meccanismi di Autenticazione HTTP sono disponibili in PHP solo quando questo viene usato come un modulo di Apache ed esso non è quindi disponibile nella versione CGI. In uno script PHP modulo di Apache, è possibile usare la funzione header() per inviare un messaggio di "Authentication Required" al browser dell'utente, provocando quindi l'apertura di una finestra contenente una richiesta di Nome utente/Password. Una volta che l'utente ha compilato i campi nome utente e password, l'URL contenente lo script PHP verrà richiamato nuovamente usando le variabili, $PHP_AUTH_USER, $PHP_AUTH_PW e $PHP_AUTH_TYPE impostate con, rispettivamente: nome, password e tipo di autenticazione. Solamente il tipo di autenticazione "Basic" è al momento supportato. Fare riferimento alla funzione header() per ulteriori informazioni.

Un frammento di script di esempio che richiede l'autenticazione da parte del browser su una pagina, potrebbe essere il seguente:

Esempio 17-1. Esempio di Autenticazione HTTP

```
<?php
  if (!isset($PHP_AUTH_USER)) {
    Header("WWW-Authenticate: Basic realm=\"Il mio Regno\"");
    Header("HTTP/1.0 401 Unauthorized");
    echo "Messaggio da inviare se si preme il tasto Cancel\n";
    exit;
  } else {
    echo "<p>Ciao $PHP_AUTH_USER.</p>";
    echo "<p>Hai inserito $PHP_AUTH_PW come tua password.</p>";
  }
?>
```

> **Note:** Fare molta attenzione quando si scrive codice per le intestazioni HTTP. Per ottenere la massima compatibilità con tutti i client, la paorla-chiave "Basic" deve essere scritta con una "B" maiuscola, la stringa realm deve essere racchiusa in virgolette doppie (non singole), ed esattamente uno spazio deve precedere il codice "401" nella linea di intestazione "HTTP/1.0 401".

Invece di stampare semplicemente $PHP_AUTH_USER e $PHP_AUTH_PW, probabilmente si vorrà controllare la validità dello username e della password. Probabilemnte inviando una chiamata al database, o cercando un utente in un file dbm.

Si faccia attenzione ad alcune versioni di Internet Explorer. Sembra che siano molto schizzinosi riguardo all'ordine delle intestazioni. Inviare l'intestazione *WWW-Authenticate* prima dell'intestazione `HTTP/1.0 401` sembra sistemare le cose per il momento.

Al fine di prevenire che qualcuno scriva uno script che rivela la password di una pagina che era stata autenticata tramite un tradizionale meccanismo esterno, le variabili PHP_AUTH non verranno impostate se è abilitata l'autenticazione esterna per quella determinata pagina. In questo caso, la variabile $REMOTE_USER può essere usata per identificare un utente autenticato esternamente.

> **Nota sulla Configurazione:** PHP usa la presenza di una direttiva `AuthType` per determinare se viene utilizzata l'autenticazione esterna. Evitare questa direttiva nel contesto dove si intende usare l'autenticazione con PHP (altrimenti ogni tentativo di autenticazione fallirà).

Si fa notare, però, che quanto scritto sopra non impedisce a qualcuno che controlla un URL non autenticato di sottrarre password da URL autenticati presenti sullo stesso server.

Sia Netscape Navigator che Internet Explorer cancellano la cache locale della finestra del browser, per quanto riguarda il realm, al ricevimento di una risposta 401 da parte del server. Questo è effettivamente un meccanismo di "log out" per l'utente, che lo forza a reinserire lo username e la password. Alcune persone usano questo per fare il "time out" dei login, o per rendere disponibili bottoni di "log-out".

Esempio 17-2. Esempio di Autenticazione HTTP che impone l'inserimento di nuovo username/password

```
<?php
  function authenticate() {
    Header( "WWW-Authenticate: Basic realm=\"Prova del Sistema di Autenticazione\"");
    Header( "HTTP/1.0 401 Unauthorized");
    echo "Per poter accedere a questa risorsa occorre inserire una coppia lo-
gin e password valide\n";
    exit;
  }

  if (!isset($PHP_AUTH_USER) || ($SeenBefore == 1 && !strcmp($OldAuth, $PHP_AUTH_USER)))
   authenticate();
  }
  else {
   echo "<p>Benvenuto: $PHP_AUTH_USER<br>";
   echo "Vecchio: $OldAuth";
   echo "<form action='$PHP_SELF' method='POST'>\n";
   echo "<input type='HIDDEN' name='SeenBefore' value='1'>\n";
   echo "<input type='HIDDEN' name='OldAuth' value='$PHP_AUTH_USER'>\n";
   echo "<input type='submit' value='Re Authenticate'>\n";
   echo "</form></p>\n";
  }
?>
```

Questo comportamento non è richiesto dallo standard di autenticazione HTTP Basic, quindi non si dovrebbe mai fare affidamento su di esso. Test effettuati con Lynx mostrano che Lynx non cancella le credenziali di autenticazione al ricevimento del codice di risposta 401 da parte del server, quindi, premendo indietro e avanti nuovamente, darà nuovamente accesso alla risorsa, ammesso che le rispettive richieste di credenziali non siano cambiate. L'utente può però premere il tasto '_' al fine di cancellare le sue informazioni di autenticazione.

Si noti anche che questo non funziona con il server IIS di Microsoft e con la versione CGI di PHP a causa di una limitazione di IIS.

Capitolo 18. Cookies

PHP supporta in modo trasparente i cookies HTTP. I cookies sono un meccanismo per memorizzare dati nel browser remoto e tenere traccia degli utenti o identificarli al loro ritorno. I cookies possono essere impostati tramite la funzione setcookie(). I cookies sono parte dell'intestazione HTTP, quindi setcookie() deve essere chiamata prima che qualsiasi output sia inviato al browser. Si tratta della stessa limitazione della funzione header(). Si può utilizzare la funzione di buffer dell'output per posticipare l'output dello script finchè non avete stabilito se impostare o meno qualsiasi cookies o l"invio di header.

Ogni cookie inviato dal client sarà automaticamente trasformato in una variabile PHP, come avviene nel caso di dati GET o POST, in base alle variabili di configurazione `register_globals` e `variables_order`. Se si vogliono assegnare più valori ad un singolo cookie, basta aggiungere *[]* al nome del cookie.

Nel PHP 4.1.0 e successivi, il vettore auto-globale `$_COOKIE` sarà sempre impostato con qualsiasi cookies inviati dal client. `$HTTP_COOKIE_VARS` è anch'essa impostata nelle versioni precedenti del PHP se è impostata la variabile di configurazione `track_vars`.

Per maggiori dettagli si veda la funzione setcookie().

Capitolo 19. Handling file uploads

POST method uploads

PHP is capable of receiving file uploads from any RFC-1867 compliant browser (which includes Netscape Navigator 3 or later, Microsoft Internet Explorer 3 with a patch from Microsoft, or later without a patch). This feature lets people upload both text and binary files. With PHP's authentication and file manipulation functions, you have full control over who is allowed to upload and what is to be done with the file once it has been uploaded.

Note that PHP also supports PUT-method file uploads as used by Netscape Composer and W3C's Amaya clients. See the PUT Method Support for more details.

A file upload screen can be built by creating a special form which looks something like this:

Esempio 19-1. File Upload Form

```
<form enctype="multipart/form-data" action="_URL_" method="post">
<input type="hidden" name="MAX_FILE_SIZE" value="1000">
Send this file: <input name="userfile" type="file">
<input type="submit" value="Send File">
</form>
```

The _URL_ should point to a PHP file. The MAX_FILE_SIZE hidden field must precede the file input field and its value is the maximum filesize accepted. The value is in bytes.

Attenzione

The MAX_FILE_SIZE is advisory to the browser. It is easy to circumvent this maximum. So don't count on it that the browser obeys you wish! The PHP-settings for maximum-size, however, cannot be fooled.

Variables defined for uploaded files differs depends on PHP version and configuration. Following variables will be defined within the destination script upon a successful upload. When track_vars is enabled, $HTTP_POST_FILES/$_FILES array is initialized. Finally, related variables may be initialized as globals when register_globals is turned on. However, use of globals is not recommended anymore.

> **Nota:** track_vars is always on from PHP 4.0.3. From PHP 4.1.0 or later, $_FILES may be used instead of `$HTTP_POST_FILES`. `$_FILES` is always global, so `global` should not be used for $_FILES in function scope.

`$HTTP_POST_FILES`/`$_FILES` is provided to contain the uploaded file information.

The contents of `$HTTP_POST_FILES` are as follows. Note that this assumes the use of the file upload name 'userfile', as used in the example script above:

`$HTTP_POST_FILES['userfile']['name']`

The original name of the file on the client machine.

`$HTTP_POST_FILES['userfile']['type']`

The mime type of the file, if the browser provided this information. An example would be `"image/gif"`.

`$HTTP_POST_FILES['userfile']['size']`

The size, in bytes, of the uploaded file.

`$HTTP_POST_FILES['userfile']['tmp_name']`

The temporary filename of the file in which the uploaded file was stored on the server.

> **Nota:** PHP 4.1.0 or later supports a short track variable `$_FILES`. PHP 3 does not support `$HTTP_POST_FILES`.

When register_globals is turned on in `php.ini` the available variables are as follows. Note that the following variable names assume the use of the file upload name 'userfile', as used in the example script above:

- `$userfile` - The temporary filename in which the uploaded file was stored on the server machine.
- `$userfile_name` - The original name or path of the file on the sender's system.
- `$userfile_size` - The size of the uploaded file in bytes.
- `$userfile_type` - The mime type of the file if the browser provided this information. An example would be "image/gif".

Note that the "`$userfile`" part of the above variables is whatever the name of the <input> field of type="file" is in the upload form. In the above upload form example, we chose to call it "userfile".

> **Nota:** `register_globals = On` is not recommended for security and performance reason.

Files will by default be stored in the server's default temporary directory, unless another location has been given with the upload_tmp_dir directive in `php.ini`. The server's default directory can be changed by setting the environment variable TMPDIR in the environment in which PHP runs. Setting it using putenv() from within a PHP script will not work. This environment variable can also be used to make sure that other operations are working on uploaded files, as well.

Esempio 19-2. Validating file uploads

The following examples are for versions of PHP 4 greater than 4.0.2. See the function entries for is_uploaded_file() and move_uploaded_file().

```
<?php
// In PHP 4.1.0 or later, $_FILES should be used instead of $HTTP_POST_FILES.
if (is_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'])) {
    copy($HTTP_POST_FILES['userfile']['tmp_name'], "/place/to/put/uploaded/file");
} else {
```

```
    echo "Possible file upload attack. Filename: " . $HTTP_POST_FILES['userfile']['name'
}
/* ...or... */
move_uploaded_file($HTTP_POST_FILES['userfile']['tmp_name'], "/place/to/put/uploaded/file
?>
```

The PHP script which receives the uploaded file should implement whatever logic is necessary for determining what should be done with the uploaded file. You can for example use the `$HTTP_POST_FILES['userfile']['size']` variable to throw away any files that are either too small or too big. You could use the `$HTTP_POST_FILES['userfile']['type']` variable to throw away any files that didn't match a certain type criteria. Whatever the logic, you should either delete the file from the temporary directory or move it elsewhere.

The file will be deleted from the temporary directory at the end of the request if it has not been moved away or renamed.

Common Pitfalls

The `MAX_FILE_SIZE` item cannot specify a file size greater than the file size that has been set in the upload_max_filesize ini-setting. The default is 2 Megabytes.

If memory limit is enabled, larger memory_limit may be needed. Make sure to set memory_limit large enough.

If max_execution_time is set too small, script execution may be exceeded the value. Make sure to set `max_execution_time` large enough.

If post_max_size set too small, large files cannot be uploaded. Make sure to set `post_max_size` large enough.

Not validating which file you operate on may mean that users can access sensitive information in other directories.

Please note that the CERN httpd seems to strip off everything starting at the first whitespace in the content-type mime header it gets from the client. As long as this is the case, CERN httpd will not support the file upload feature.

Uploading multiple files

Multiple files can be uploaded using different `name` for `input`.

It is also possible to upload multiple files simultaneously and have the information organized automatically in arrays for you. To do so, you need to use the same array submission syntax in the HTML form as you do with multiple selects and checkboxes:

Nota: Support for multiple file uploads was added in version 3.0.10.

Esempio 19-3. Uploading multiple files

```
<form action="file-upload.php" method="post" enctype="multipart/form-data">
  Send these files:<br>
  <input name="userfile[]" type="file"><br>
  <input name="userfile[]" type="file"><br>
  <input type="submit" value="Send files">
</form>
```

When the above form is submitted, the arrays `$HTTP_POST_FILES['userfile']`, `$HTTP_POST_FILES['userfile']['name']`, and `$HTTP_POST_FILES['userfile']['size']` will be initialized. (As well as in $_FILES for PHP 4.1.0 or later. $HTTP_POST_VARS in PHP 3. When `register_globals` is on, globals for uploaded files are also initialized). Each of these will be a numerically indexed array of the appropriate values for the submitted files.

For instance, assume that the filenames `/home/test/review.html` and `/home/test/xwp.out` are submitted. In this case, `$HTTP_POST_FILES['userfile']['name'][0]` would contain the value `review.html`, and `$HTTP_POST_FILES['userfile']['name'][1]` would contain the value `xwp.out`. Similarly, `$HTTP_POST_FILES['userfile']['size'][0]` would contain `review.html`'s filesize, and so forth.

`$HTTP_POST_FILES['userfile']['name'][0]`, `$HTTP_POST_FILES['userfile']['tmp_name'][0]`, `$HTTP_POST_FILES['userfile']['size'][0]`, and `$HTTP_POST_FILES['userfile']['type'][0]` are also set.

PUT method support

PHP provides support for the HTTP PUT method used by clients such as Netscape Composer and W3C Amaya. PUT requests are much simpler than a file upload and they look something like this:

```
PUT /path/filename.html HTTP/1.1
```

This would normally mean that the remote client would like to save the content that follows as: /path/filename.html in your web tree. It is obviously not a good idea for Apache or PHP to automatically let everybody overwrite any files in your web tree. So, to handle such a request you have to first tell your web server that you want a certain PHP script to handle the request. In Apache you do this with the *Script* directive. It can be placed almost anywhere in your Apache configuration file. A common place is inside a <Directory> block or perhaps inside a <Virtualhost> block. A line like this would do the trick:

```
Script PUT /put.php
```

This tells Apache to send all PUT requests for URIs that match the context in which you put this line to the put.php script. This assumes, of course, that you have PHP enabled for the .php extension and PHP is active.

Inside your put.php file you would then do something like this:

```
<?php copy($PHP_UPLOADED_FILE_NAME,$DOCUMENT_ROOT.$REQUEST_URI); ?>
```

This would copy the file to the location requested by the remote client. You would probably want to perform some checks and/or authenticate the user before performing this file copy. The only trick here is that when PHP sees a PUT-method request it stores the uploaded file in a temporary file just like those handled but the POST-method. When the request ends, this temporary file is deleted. So, your PUT handling PHP script has to copy that file somewhere. The filename of this temporary file is in the $PHP_PUT_FILENAME variable, and you can see the suggested destination filename in the $REQUEST_URI (may vary on non-Apache web servers). This destination filename is the one that the remote client specified. You do not have to listen to this client. You could, for example, copy all uploaded files to a special uploads directory.

Capitolo 20. Utilizzo di file remoti

Quando viene abilitato il supporto per l "URL fopen wrapper" durante la configurazione di PHP (avviene automaticamente a meno che si specifichi espressamente il flag `--disable-url-fopen-wrapper` (per le versioni sino alla 4.0.3) oppure impostare *`allow_url_fopen`* a off in `php.ini` (per le nuove versioni)), si possono usare URL FTP e HTTP con la maggior parte delle funzioni che richiedono nomi di file come parametri, incluse le funzioni require() e include().

> **Nota:** Non si possono usare i file remoti con include() e require() sotto Windows.

Per esempio, si può usare per aprire un file da un web server remoto, elaborare i dati presi da remoto, e usarli per effetuare delle query, o semplicemente visualizzarli con lo stile del proprio sito web.

Esempio 20-1. Legere il titolo di una pagina web remota

```
<?php
$file = fopen ("http://www.example.com/", "r");
if (!$file) {
    echo "<p>Impossibile aprire il file remoto.\n";
    exit;
}
while (!feof ($file)) {
    $linea = fgets ($file, 1024);
    /* Funziona solo se title e i relativi tag sono sulla medesima riga */
    if (eregi ("<title>(.*)</title>", $linea, $out)) {
        $title = $out[1];
        break;
    }
}
fclose($file);
?>
```

Si può anche scrivere in un file remoto via FTP se l'utente con cui ci si connette ha le autorizzazioni necessarie, e il file non è già presente. Per connettersi con un utenti specifico si ha bisogno di specificare lo username (e la relativa password) dentro l'URL in questo modo: 'ftp://user:password@ftp.example.com/directory/del/file'. (Si può usare lo stesso tipo di sintassi per accedere a file via HTTP quando richiedono autenticazione).

Esempio 20-2. Salvataggio di dati su server remoto

```
<?php
$file = fopen ("ftp://ftp.example.com/incoming/outputfile", "w");
if (!$file) {
    echo "<p>Impossibile aprire il file remoto in scrittura.\n";
    exit;
}
/* Scrive i dati qui. */
```

```
fputs ($file, $HTTP_SERVER_VARS['HTTP_USER_AGENT'] . "\n");
fclose ($file);
?>
```

Nota: Dall'esempio precedente ci si può fare un'idea di come usare questa tecnica per effettuare dei log in remoto, ma come già accennato non è possibile scrivere su file già esistenti con questo sitema. Per fare una procedura di log distribuito è più indicata la funzione syslog().

Capitolo 21. Connection handling

> **Nota:** The following applies to 3.0.7 and later.

Internally in PHP a connection status is maintained. There are 3 possible states:

- 0 - NORMAL
- 1 - ABORTED
- 2 - TIMEOUT

When a PHP script is running normally the NORMAL state, is active. If the remote client disconnects the ABORTED state flag is turned on. A remote client disconnect is usually caused by the user hitting his STOP button. If the PHP-imposed time limit (see set_time_limit()) is hit, the TIMEOUT state flag is turned on.

You can decide whether or not you want a client disconnect to cause your script to be aborted. Sometimes it is handy to always have your scripts run to completion even if there is no remote browser receiving the output. The default behaviour is however for your script to be aborted when the remote client disconnects. This behaviour can be set via the ignore_user_abort `php.ini` directive as well as through the corresponding "php_value ignore_user_abort" Apache .conf directive or with the ignore_user_abort() function. If you do not tell PHP to ignore a user abort and the user aborts, your script will terminate. The one exception is if you have registered a shutdown function using register_shutdown_function(). With a shutdown function, when the remote user hits his STOP button, the next time your script tries to output something PHP will detect that the connection has been aborted and the shutdown function is called. This shutdown function will also get called at the end of your script terminating normally, so to do something different in case of a client disconnect you can use the connection_aborted() function. This function will return `TRUE` if the connection was aborted.

Your script can also be terminated by the built-in script timer. The default timeout is 30 seconds. It can be changed using the max_execution_time `php.ini` directive or the corresponding "php_value max_execution_time" Apache .conf directive as well as with the set_time_limit() function. When the timer expires the script will be aborted and as with the above client disconnect case, if a shutdown function has been registered it will be called. Within this shutdown function you can check to see if a timeout caused the shutdown function to be called by calling the connection_timeout() function. This function will return `TRUE` if a timeout caused the shutdown function to be called.

One thing to note is that both the ABORTED and the TIMEOUT states can be active at the same time. This is possible if you tell PHP to ignore user aborts. PHP will still note the fact that a user may have broken the connection, but the script will keep running. If it then hits the time limit it will be aborted and your shutdown function, if any, will be called. At this point you will find that connection_timeout() and connection_aborted() return `TRUE`. You can also check both states in a single call by using the connection_status(). This function returns a bitfield of the active states. So, if both states are active it would return 3, for example.

Capitolo 22. Connessioni Persistenti ai Database

Connessioni persistenti sono collegamenti SQL che non vengono chiusi quando l'esecusione di uno script viene terminata. Quando è richiesta una connessione persistente, PHP controlla se esiste già una identica connessione persistente (che è rimasta aperta da prima) - e se esiste, la usa. Se non esiste, crea il collegamento. Una connessione 'identica' è una connessione aperta verso lo stesso host, con lo stesso username e la stessa password (dove applicabile).

> **Nota:** Ci sono altre estensioni che permettono di usare connessioni persistenti, ad esempio l'estensione IMAP.

Chi non ha molta familiarità con il modo in cui lavorano i server web e di come essi distribuiscano il carico potrebbero confondere le connessioni permanenti per ciò che esse non sono. In particolare, *non* danno la possibilità di aprire 'sessioni utente' sullo stesso collegamento SQL, *non* danno la possibilità di gestire una transazione in maniera efficiente e sopratutto non fanno molte altre cose. Infatti, per essere molto chiari su questo punto, le connessioni persistenti non hanno *nessuna* funzionalià che non era disponibile con le loro omonime non-persistenti.

Perché?

Questo ha a che fare con il modo in cui i server web operano. Ci sono tre modi in cui un server web può utilizzare PHP per generare le pagine web.

Il primo metodo ' quello di usare il PHP come un "wrapper" (involucro) CGI. Quando viene eseguito in questa modalità, per ogni pagina che viene richiesta al server web che contenga del codice PHP, viene creata e, alla fine dell'esecuzione, distrutta, una istanza dell'interprete PHP. Poiché viene distrutta alla fine di ogni richiesta, qualunque risorsa abbia acquisito (come ad esempio un collegamento ad un server di database SQL) verrà anch'essa distrutta. In questo caso, non vi è nessun guadagno nell'usare le connessioni persistenti -- semplicemente esse non persistono.

Il secondo, e più popolare, metodo è quello di eseguire il programma PHP come modulo in un server web multiprocesso, cosa che attualmente include solo Apache. Un server multiprocesso ha tipicamente un processo (il padre) che coordina un insieme di processi (i suoi figli) che sono quelli che generano le pagine web. Quando arriva una richiesta da parte di un client, questa è passata ad uno dei figli che in quel momento non è già occupato a servire un altro client. Questo significa che quando lo stesso client effettua una seconda richiesta al server, esso potrà essere servito da un diverso processo figlio rispetto a quello della prima volta. In questa situazione, usare una connessione persistente, permette di stabilire una e una sola connessione al database SQL per ogni processo figlio, poiché ciascun processo figlio necessita di collegarsi al database SQL solo la prima volta che richiama una pagina che ne fa uso. Quando viene richamata un'altra pagina che necessita di una connessione al server SQL, essa può riutilizzare la connessione che lo stesso processo figlio aveva stabilito in precedenza.

L'ultimo metodo è quello di usare il PHP come una plug-in per un server web multithread. Attualmente PHP 4 supporta ISAPI, WSAPI e NSAPI (su piattaforma Windows), i quali permettono di usare PHP come una plug-in sui server web multithread come FastTrack (iPlanet) di Netscape, Internet Information Server (IIS) di Microsoft e WebSite Pro di O'Reilly. Il comportamento è essenzialmente lo stesso che si ha nel modello multiprocesso descritto prima. Si noti che il supporto SAPI non è disponibile in PHP 3.

Se le connessioni persistenti non hanno nessuna funzionalità aggiuntiva, perch´ usarle?

La risposta, in questo caso è estremamente semplice: efficienza. Le connessioni persistenti sono utili se il carico di lavoro necessario per aprire una connessione al server SQL è alto. Che il carico sia molto alto o meno dipende da molti fattori. Quali, il tipo di database che si utilizza, il fatto che esso si trovi sulla stessa macchina sulla quale si trova il server web, quanto carico di lavoro ha la macchina sulla quale si trova il server SQL, e molte altre ragioni. Il fatto importante è che se il carico

di lavoro necessario per aprire le connessioni è alto, le connessioni persistenti possono aiutare in maniera considerevole. Fanno in modo che il processo figlio si colleghi semplicemente una volta durante la sua intera vita, invece di collegarsi ogni volta che processa una pagina che richiede una connessione al server SQL. Questo significa che per ogni figlio che ha aperto una connessione persistente, si avrà una nuova connessione persistente aperta verso il server. Per esempio, se si hanno 20 diversi processi figlio che eseguono uno script che crea una connessione persistente al server SQL server, si avranno 20 diverse connessioni al server SQL, una per ogni figlio.

Si fa notare, tuttavia, che questo può avere degli svantaggi se si fa uso di un database che ha un limite al numero di connessioni, minore rispetto al numero delle connessioni persistenti dei procesi figlio. Se per esempio di usa un database con 16 connessioni simultanee, e durante un periodo di intensa attività del web server, 17 processi figlio cercano di collegarsi, uno non sarà in grado di farlo. Se nei vostri script ci sono bug che non permettono alle connessioni di chiudersi in maniera regolare (come ad esempio un loop infinito), un database con sole 32 connessioni diventerà rapidamente saturo. Fare riferimento alla documentazione del database per informazioni su come gestire connessioni abbandonate o inattive.

Attenzione

Ci sono un paio di altri caveat da tenere in considerazione quando si usano le connessioni persistenti. Uno è che quando si usa il table locking su una connessione persistente, se lo script, per una qualunque ragione non è in grado di rimuovere il blocco, allora i successivi script che useranno la stessa connessione rimarranno bloccati in maniera indefinita e potrebbe essre necessario riavviare il server httpd o il server database. Un altro è che quando si usano le transazioni, un transaction block verrà trasferito anche allo script successivo che usa la medesima connessione, se lo script in esecuzione termina prima che sia terminato il transaction block. In entrambi i casi, si può usare register_shutdown_function() per registrare una semplice funzione di pulizia per sbloccare le tabelle o effettuare il roll back delle transaczioni. Sarebbe meglio non dover affrontare il problema, semplicemente non usando le connessioni persistenti negli script che usano i lock delle tabelle o le transaczioni (si possono comunque usare in altre situazioni).

Sommario importante. Le connessioni persistenti sono state pensate per avere una mappatura uno-a-uno con le connessioni di tipo normale. Ciò significa che si dovrebbe *sempre* essere in grado di cambiare una connessione persistente con una connessione non-persistente, e che questo non dovrebbe cambiare il modo in cui lo script si comporta. *Può* (e probabilmente succederà) cambiare l'efficienza dello script, ma non il suo comportamento!

Vedere anche fbsql_pconnect(), ibase_pconnect(), ifx_pconnect(), imap_popen(), ingres_pconnect(), msql_pconnect(), mssql_pconnect(), mysql_pconnect(), **OCIPLogon()**, odbc_pconnect(), **Ora_pLogon()**, pfsockopen(), pg_pconnect() e sybase_pconnect().

Capitolo 23. Modalità sicura (Safe mode)

La modalità Safe Mode è un tentativo di risolvere il problema di sicurezza derivante dalla condivisione del server. Dal punto di vista architetturale non è corretto cercare di risolvere questo problema al livello del PHP, ma poiché le alternative al livello del web server e del SO (Sistema Operativo) non sono realistiche, in molti, specialmente ISP (Internet Service Provider), utilizzano la modalità sicura.

Le direttive di configurazione che controllano la modalità sicura sono:

```
safe_mode = Off
open_basedir =
safe_mode_exec_dir =
safe_mode_allowed_env_vars = PHP_
safe_mode_protected_env_vars = LD_LIBRARY_PATH
disable_functions =
```

Quando safe_mode è attiva (on), il PHP verifica se il proprietario dello script in esecuzione e il proprietario del file su cui si sta operando con una funzione sui file, coincidono. Per esempio:

```
-rw-rw-r--    1 rasmus   rasmus       33 Jul  1 19:20 script.php
-rw-r--r--    1 root     root       1116 May 26 18:01 /etc/passwd
```

Eseguendo questo script.php

```
<?php
 readfile('/etc/passwd');
?>
```

results in this error when Safe Mode is enabled:

```
Warning: SAFE MODE Restriction in effect. The script whose uid is 500 is not
allowed to access /etc/passwd owned by uid 0 in /docroot/script.php on line 2
```

Se, invece di safe_mode, viene definita una directory open_basedir allora tutte le operazioni sui file saranno limitate ai file sottostanti la directory specificata. Per esempio (nel file httpd.conf di Apache):

```
<Directory /docroot>
  php_admin_value open_basedir /docroot
</Directory>
```

If you run the same script.php with this open_basedir setting then this is the result:

```
Warning: open_basedir restriction in effect. File is in wrong directory in
/docroot/script.php on line 2
```

È possibile inoltre disabilitare le singole funzioni. Se si aggiunge la seguente riga al file php.ini:

```
disable_functions readfile,system
```

Then we get this output:

```
Warning: readfile() has been disabled for security reasons in
/docroot/script.php on line 2
```

Funzioni limitate/disabilitate dalla modalità sicura (safe-mode)

Questo è un elenco probabilmente ancora incompleto e forse non esatto delle funzioni limitate da Safe Mode.

Tabella 23-1. Funzioni limitate dalla modalità sicura

Funzioni	Limitazioni
dbmopen()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
dbase_open()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
filepro()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
filepro_rowcount()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
filepro_retrieve()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
ifx_*()	sql_safe_mode restrictions, (!= Safe Mode)
ingres_*()	sql_safe_mode restrictions, (!= Safe Mode)

Funzioni	**Limitazioni**
mysql_*()	sql_safe_mode restrictions, (!= Safe Mode)
pg_loimport()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
posix_mkfifo()	Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione.
putenv()	Obbedisce le direttive del file ini safe_mode_protected_env_vars e safe_mode_allowed_env_vars. Vedere la documentazione relativa on putenv()
move_uploaded_file()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
chdir()	Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione.
dl()	Questa funzione è disabilitata nella modalitàsafe-mode
backtick operator	Questa funzione è disabilitata nella modalitàsafe-mode
shell_exec() (functional equivalent of backticks)	Questa funzione è disabilitata nella modalitàsafe-mode
exec()	You can only execute executables within the safe_mode_exec_dir. For practical reasons it's currently not allowed to have `..` components in the path to the executable.
system()	You can only execute executables within the safe_mode_exec_dir. For practical reasons it's currently not allowed to have `..` components in the path to the executable.
passthru()	You can only execute executables within the safe_mode_exec_dir. For practical reasons it's currently not allowed to have `..` components in the path to the executable.
popen()	You can only execute executables within the safe_mode_exec_dir. For practical reasons it's currently not allowed to have `..` components in the path to the executable.
mkdir()	Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione.
rmdir()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.

Funzioni	**Limitazioni**
rename()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione. Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione.
unlink()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione. Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione.
copy()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione. Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione. (on *`source`* and *`target`*)
chgrp()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
chown()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione.
chmod()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione. In addition, you cannot set the SUID, SGID and sticky bits
touch()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione. Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione.
symlink()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione. Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione. (note: only the target is checked)
link()	Controlla che i file o le directory su cui si sta lavorando, abbiano lo stesso UID dello script che è in esecuzione. Controlla che la directory su cui si sta lavorando, abbia lo stesso UID dello script che è in esecuzione. (note: only the target is checked)
getallheaders()	In Safe Mode, headers beginning with 'authorization' (case-insensitive) will not be returned. Warning: this is broken with the aol-server implementation of getallheaders()!
Qualsiasi funzione che utilizza `php4/main/fopen_wrappers.c`	??

Capitolo 24. Using PHP from the command line

Since version 4.3, PHP supports a new SAPI type (Server Application Programming Interface) named CLI which means *Command Line Interface*. As the name implies, this SAPI type main focus is on developing shell (or desktop as well) applications with PHP. There are quite some differences between the CLI SAPI and other SAPIs which are further explained throughout this chapter.

The CLI SAPI was released for the first time with PHP 4.2.0, but was still experimental back then and had to be explicitely enabled with --enable-cli when running ./configure. Since PHP 4.3.0 the CLI SAPI is no longer experimental and is therefore **always** built and installed as the php (called php.exe on Windows) binary.

Remarkable differences of the CLI SAPI compared to other SAPIs:

- Unlikely the CGI SAPI, no headers are written to the output.

  Though the CGI SAPI provies a way to suppress HTTP headers, there's not equivalent switch to enable them in the CLI SAPI.

- The are certain php.ini directives which are overriden by the CLI SAPI because the do not make sense in shell environments:

Tabella 24-1. Overriden php.ini directives

Directive	CLI SAPI default value	Comment
html_errors	FALSE	It can be quite hard to read the error message in your shell when it's cluttered with all those meaningless HTML tags, therefore this directive defaults to FALSE.
implicit_flush	TRUE	It is desired that any output coming from print(), echo() and friends is immidiately written to the output and not cached in any buffer. You still can use output buffering if you want to defer or manipulate standard output.
max_execution_time	0 (unlimited)	Due to endless possibilities of using PHP in shell environments, the maximum execution time has been set to unlimited. Whereas applications written for the web are executed within splits of a seconds, shell application tend to have a much longer execution time.
register_argc_argv	TRUE	The global PHP variables $argc (number of arguments passed to the application) and $argv (array of the actual arguments) are always registered and filled in with the appropriate values when using the CLI SAPI.

> **Nota:** This directives cannot be initialzied with another value from the configuration file `php.ini` or a custom one (if specified). This is a limitation because those default values are applied after all configuration files have been parsed. However, their value can be changed during runtime (which does not make sense for all of those directives, e.g. register_argc_argv).

- The `CLI SAPI` does **not** change the current directory to the directory of the executed script !

  Example showing the difference to the `CGI SAPI`:

```
<?
    /* Our simple test application */
    echo getcwd(), "\n";
?>
```

  When using the `CGI` version, the output is

```
$ pwd
/tmp

$ php-cgi -f another_directory/test.php
/tmp/another_directory
```

  This clearly shows that `PHP` changes its current directory to the one of the executed script.

  Using the `CLI SAPI` yields:

```
$ pwd
/tmp

$ php -f another_directory/test.php
/tmp
```

  This allows greater flexibility when writing shell tools in `PHP`.

  > **Nota:** The `CGI SAPI` supports the `CLI SAPI` behaviour by means of the `-C` switch when ran from the command line.

The list of command line options provided by the `PHP` binary can be queryied anytime by running `PHP` with the `-h` switch:

```
Usage: php [options] [-f] <file> [args...]
       php [options] -r <code> [args...]
```

```
     php [options] [-- args...]
  -s               Display colour syntax highlighted source.
  -w               Display source with stripped comments and whitespace.
  -f <file>        Parse <file>.
  -v               Version number
  -c <path>|<file> Look for php.ini file in this directory
  -a               Run interactively
  -d foo[=bar]     Define INI entry foo with value 'bar'
  -e               Generate extended information for debugger/profiler
  -z <file>        Load Zend extension <file>.
  -l               Syntax check only (lint)
  -m               Show compiled in modules
  -i               PHP information
  -r <code>        Run PHP <code> without using script tags <?..?>
  -h               This help

  args...          Arguments passed to script. Use -- args when first argument
                   starts with - or script is read from stdin
```

The `CLI SAPI` has three different ways of getting the `PHP` code you want to execute:

1. Telling `PHP` to execute a certain file.

   ```
   php my_script.php

   php -f my_script.php
   ```

   Both ways (using the `-f` switch or not) execute the given file `my_script.php`. You can choose any file to execute, your `PHP` scripts do not have to end with the `.php` extension but can give them any name or extension you want them to have.

2. Pass the `PHP` code to execute directly on the command line.

   ```
   php -r 'print_r(get_defined_constants());'
   ```

   Special care has to be taken in regards of shell variable substitution and quoting usage.

   Nota: Read the example carefully, thera are no beginning or ending tags! The `-r` switch simply does not need them. Using them will lead to a parser error.

3. Provide the `PHP` code to execute via standard input (`stdin`).

   This gives the powerful ability to dynamically create `PHP` code and feed it to the binary, as shown in this (fictional) example:

```
$ some_application | some_filter | php | sort -u >final_output.txt
```

You cannot combine any of the three ways to execute code.

Like every shell application not only the PHP binary accepts a number of arguments but also your PHP script can receive them. The number of arguments which can be passed to your script is not limited by PHP (the shell has a certain size limit in numbers of characters which can be passed; usually you won't hit this limit). The arguments passed to your script are available in the global array `$argv`. The zero index always contains the script name (which is `-` in case the PHP code is coming from either standard input or from the command line switch `-r`). The second registered global variable is `$argc` which contains the number of elements in the `$argv` array (**not** the number of arguments passed to the script).

As long as the arguments you want to pass to your script do not start with the `-` character, there's nothing special to watch out for. Passing an argument to your script which starts with a `-` will cause trouble because PHP itself thinks it has to handle it. To prevent this use the argument list separator `--`. After the argument has been parsed by PHP, every argument following it is passed untoched/unparsed to your script.

```
# This will not execute the given code but will show the PHP usage
$ php -r 'var_dump($argv);' -h
Usage: php [options] [-f] <file> [args...]
[...]

# This will pass the '-h' argument to your script and prevent PHP from show-
ing it's usage
$ php -r 'var_dump($argv);' -- -h
array(2) {
  [0]=>
  string(1) "-"
  [1]=>
  string(2) "-h"
}
```

However, there's another way of using PHP for shell scripting. You can write a script which's first line starts with `#!/usr/bin/php` and then following the normal PHP code included within the PHP starting and end tags and set the execution attributes of the file appropriately. This way it can be executed like a normal shell or perl script:

```
#!/usr/bin/php
<?
    var_dump($argv);
?>
```

Assuming this file is named `test` in the current directory, we can now do the following:

```
$ chmod 755 test
$ ./test -h -- foo
array(4) {
  [0]=>
  string(6) "./test"
  [1]=>
  string(2) "-h"
  [2]=>
  string(2) "--"
  [3]=>
  string(3) "foo"
}
```

As you see no care has to be taken when passing parameters to your script which start with `-`.

Tabella 24-2. Command line options

Option	Description
-s	Display colour syntax highlighted source. This option uses the internal mechanism to parse the file and produces a `HTML` highlighted version of it and writes it to standard output. Note that all it does it to generate a block of `<code> [...] </code>` `HTML` tags, no `HTML` headers. **Nota:** This option does not work together with the `-r` option.
-w	Display source with stripped comments and whitespace. **Nota:** This option does not work together with the `-r` option.
-f	Parses and executed the given filename to the `-f` option. This switch is optional and can be left out. Only providing the filename to execute is sufficient.

-v	Writes the PHP, PHP SAPI, and Zend version to standard output, e.g.
	
	`$ php -v`
	`PHP 4.3.0-dev (cli), Copyright (c) 1997-2002 The PHP Group`
	`Zend Engine v1.2.1, Copyright (c) 1998-2002 Zend Technologies`

Option	**Description**

-c	With this option one can either specify a directory where to look for `php.ini` or you can specify a custom `INI` file directly (which does not need to be named `php.ini`), e.g.:
	
	`$ php -c /custom/directory/ my_script.php`
	
	`$ php -c /custom/directory/custom-file.ini my_scrip`

-a	Runs PHP interactively.

-d	This option allows to set a custom value for any of the configuration directives allowed in `php.ini`. The syntax is:
	
	`-d configuration_directive[=value]`
	
	Examples:
	
	`# Ommiting the value part will set the given con-figuration directive to "1"`
	`$ php -d max_execution_time -r '$foo = ini_get("max`
	`string(1) "1"`
	
	`# Passing an empty value part will set the con-figuration directive to ""`
	`php -d max_execution_time= -r '$foo = ini_get("max_`
	`string(0) ""`
	
	`# The configuration directive will be set to any-thing passed after the '=' character`
	`$  php -d max_execution_time=20 -r '$foo = ini_get(`
	`string(2) "20"`
	`$  php -d max_execution_time=doesntmakesense -r '$foo = ini_get("max_execution_time"); var_dump($`
	`string(15) "doesntmakesense"`

-e	Generate extended information for debugger/profiler.

Option	Description
-z	Load Zend extension. If only a filename is given, PHP tries to load this extension from the current default library path on your system (usually specified `/etc/ld.so.conf` on Linux systems). Passing a filename with an absolute path information will not use the systems library search path. A relative filename with a directory information will tell `PHP` only to try to load the extension relative to the current directory.
-l	This option provides a convenient way to only perform a syntax check on the given `PHP` code. On succes, the text `No syntax errors detected in <filename>` is written to standard output and the shell return code is `0`. On failure, the text `Errors parsing <filename>` in addition to the internal parser error message is written to standard output and the shell return code is set to `255`. This option won't find fatal errors (like undefined functions). Use `-f` if you would like to test for fatal errors too. **Nota:** This option does not work together with the `-r` option.

-m	Using this option, PHP prints out the built in (and loaded) PHP and Zend modules:
	
	`$ php -m`
	`[PHP Modules]`
	`xml`
	`tokenizer`
	`standard`
	`session`
	`posix`
	`pcre`
	`overload`
	`mysql`
	`mbstring`
	`ctype`
	
	`[Zend Modules]`

-i	This command line option calls phpinfo(), and prints out the results. If `PHP` is not working well, it is advisable to make a `php -i` and see if any error messages are printed out before or in place of the information tables. Beware that the output is in `HTML` and therefore quite huge.

Option	Description

-r	This option allows execution of PHP right from within the command line. The PHP start and end tags (`<?php` and `?>`) are **not needed** and will cause a parser errors.
	
	Nota: Care has to be taken when using this form of PHP to not collide with command line variable substitution done by the shell.
	
	Example showing a parser error
	
	`$ php -r "$foo = get_defined_constants();"`
	`Command line code(1) : Parse error - parse er-`
	`ror, unexpected '='`
	
	The problem here is that the sh/bash performs variable substritution even when using double quotes `"`. Since the variable `$foo` is unlikely to be defined, it expands to nothing which results in being the code passed to PHP for executin in fact reads:
	
	`$ php -r " = get_defined_constants();"`
	
	The correct way would be to use single quotes `'`. variables in strings quoted with single quotes are not expanded by sh/bash.
	
	`$ php -r '$foo = get_defined_constants(); var_dump($`
	`array(370) {`
	`  ["E_ERROR"]=>`
	`  int(1)`
	`  ["E_WARNING"]=>`
	`  int(2)`
	`  ["E_PARSE"]=>`
	`  int(4)`
	`  ["E_NOTICE"]=>`
	`  int(8)`
	`  ["E_CORE_ERROR"]=>`
	`  [...]`
	
	If you are using a shell different from sh/bash, you might experience further issues. Feel free to open a bug report or send a mail to phpdoc@lists.php.net. One still can easily run intro troubles when trying to get shell variables into the code or using backslashes for escaping. You've been warned.

-h	With this option, you can get information about the actual list of command line options and some one line descriptions about what they do.

The PHP executable can be used to run PHP scripts absolutely independent from the web server. If you are on a Unix system, you should add a special first line to your PHP script, and make it executable, so the system will know, what program should run the script. On a Windows platform you can associate `php.exe` with the double click option of the `.php` files, or you can make a batch file to run the script through PHP. The first line added to the script to work on Unix won't hurt on Windows, so you can write cross platform programs this way. A simple example of writing a command line PHP program can be found below.

Esempio 24-1. Script intended to be run from command line (script.php)

```
#!/usr/bin/php
<?php

if ($argc != 2 || in_array($argv[1], array('--help', '-help', '-h', '-?'))) {
?>

This is a command line PHP script with one option.

  Usage:
  <?php echo $argv[0]; ?> <option>

  <option> can be some word you would like
  to print out. With the --help, -help, -h,
  or -? options, you can get this help.

<?php
} else {
    echo $argv[1];
}
?>
```

In the script above, we used the special first line to indicate, that this file should be run by PHP. We work with a CLI version here, so there will be no HTTP header printouts. There are two variables you can use while writing command line applications with PHP: `$argc` and `$argv`. The first is the number of arguments plus one (the name of the script running). The second is an array containing the arguments, starting with the script name as number zero (`$argv[0]`).

In the program above we checked if there are less or more than one arguments. Also if the argument was `--help`, `-help`, `-h` or `-?`, we printed out the help message, printing the script name dynamically. If we received some other argument we echoed that out.

If you would like to run the above script on Unix, you need to make it executable, and simply call it as `script.php echothis` or `script.php -h`. On Windows, you can make a batch file for this task:

Esempio 24-2. Batch file to run a command line PHP script (script.bat)

```
@c:\php\php.exe script.php %1 %2 %3 %4
```

Assuming, you named the above program as `script.php`, and you have your `php.exe` in `c:\php\php.exe` this batch file will run it for you with your added options: `script.bat echothis` or `script.bat -h`.

See also the Readline extension documentation for more functions you can use to enhance your command line applications in PHP.

Parte IV. Guida Funzioni

I. Funzioni Apache

Queste funzioni sono disponibili unicamente quando PHP è eseguito come modulo di Apache.

apache_child_terminate (PHP 4 >= 4.0.5)

Interrompe il processo apache dopo la presente richiesta

bool **apache_child_terminate** (void) \linebreak

apache_child_terminate() informa il processo Apache che sta eseguendo la richiesta PHP corrente di terminare quando l'esecuzione del codice PHP è stata completata. Può essere usata per interrompere un processo dopo che sia stato eseguito uno script con alta occupazione di memoria dal momento che la memoria viene normalmente liberata internamente ma non restituita al sistema operativo.

> **Nota:** La disponibilità di questa caratteristica è controllata dalal direttiva del `php.ini` `apache.child_terminate`, che è impostata a `off` di default.
>
> Questa caratteristica non è inoltre disponibile sulle versioni multithread di apache come, ad esempio, la versione win32.

Vedere anche exit().

apache_lookup_uri (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Esegue una richiesta parziale della URI specificata e restituisce tutte le informazioni

object **apache_lookup_uri** (string nomefile) \linebreak

Questa funzione esegue una richiesta parziale per una URI. Esegue l'operazione finché ottiene tutte le informazioni importanti sulla risorsa e restituisce queste informazioni in una classe. Le proprietà della classe restituista sono:
status
the_request
status_line
method
content_type
handler
uri
filename
path_info
args
boundary
no_cache
no_local_copy
allowed
send_bodyct
bytes_sent
byterange
clength
unparsed_uri
mtime
request_time

> **Nota: apache_lookup_uri()** funziona solo quando PHP è installato come modulo Apache.

apache_note (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Ricava o imposta una variabile nella tabella notes di Apache

string **apache_note** (string nome_nota [, string valore]) \linebreak

apache_note() è una funzione specifica di Apache che ricava o imposta un valore nella tabella `notes` di una richiesta HTTP. Se viene invocata con un solo argomento restituisce il valore della nota `nome_nota`. Se viene chiamata con due argomenti, imposta il valore della nota `nome_nota` a `valore` e restituisce il valore precedente della nota `nome_nota`.

apache_setenv (PHP 4 >= 4.2.0)

Imposta una variabile Apache subprocess_env

int **apache_setenv** (string varibile, string valore [, bool vai_in_cima]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ascii2ebcdic (PHP 3>= 3.0.17)

Traduce una stringa da ASCII a EBCDIC

int **ascii2ebcdic** (string ascii_str) \linebreak

ascii2ebcdic() è una funzione Apache che è disponibile solo su sistemi operativi basati sull'EBCDIC (OS/390, BS2000). Traduce la stringa codificata in ASCII *`ascii_str`* nella sua rappresentazione equivalente EBCDIC (proteggendo i dati binari), e restituisce il risultato.

Vedere anche la funzione duale ebcdic2ascii()

ebcdic2ascii (PHP 3>= 3.0.17)

Traduce una stringa da string EBCDIC ad ASCII

int **ebcdic2ascii** (string ebcdic_str) \linebreak

ebcdic2ascii() è una funzione Apache che è disponibile solo su sistemi operativi basati su EBCDIC (OS/390, BS2000). Traduce la stringa codificata in EBCDIC *`ebcdic_str`* nella sua rappresentazione equivalente ASCII (proteggendo i dati binari), e restituisce il risultato.

Vedere anche la funzione duale ascii2ebcdic()

getallheaders (PHP 3, PHP 4 >= 4.0.0)

Estrae tutti gli header della richiesta HTTP

array **getallheaders** (void) \linebreak

Questa funzione restituisce un array associativo di tutti gli header HTTP nella richiesta corrente.

> **Nota:** Si può anche estrarre il valore delle variabili comuni CGI leggendole dall'ambiente, il che funziona sia che si stia usando PHPcome modulo Apache che come CGI. Usare phpinfo() per vedere una lsta di tutte le variabili d'ambiente definite in questo modo.

Esempio 1. esempio di getallheaders()

```
$headers = getallheaders();
while (list ($header, $valore) = each ($headers)) {
    echo "$header: $valore<br />\n";
}
```

Questo esempio mostrerà tutti gli header della richiesta corrente.

> **Nota: getallheaders()** è attualmente supportato solo quando PHP è eseguito come modulo Apache.

virtual (PHP 3, PHP 4 >= 4.0.0)

Esegue una sotto-richiesta Apache

int **virtual** (string nomefile) \linebreak

virtual() è una funzione specifica Apache che è equivalente a <!--#include virtual...--> in mod_include. Esegue una sotto-richiesta Apache. È utile ad includere script CGI o file .shtml, o qualsiasi altra cosa si voglia fa analizzarea ad Apache. Si noti che per uno script CGI, questo deve generare degli header CGI validi. Quindi, Come minimo deve generare un header Content-type. Nel caso di file PHP, si deve usare include() o require(); **virtual()** non può essere usata per includere un documento che deve essere interpretato come file PHP.

Al fine di eseguire la sotto-richiesta, tutti i buffer vengono chiusi e svuotati verso il browser, e anche gli header in attesa vengono inviati.

II. Funzioni di Array

Introduzione

Queste funzioni permettono di manipolare e interagire con gli array in vari modi. Gli array sono indispensabili per immagazzinare, mantenere e operare su gruppi di variabili.

Sono supportati sia array semplici che multi-dimensionali, che possono essere sia creati dall'utente che da funzioni. Ci sono specifiche funzioni di database per riempire gli array a partire da interrogazioni sui dati, e parecchie funzioni restituiscono array.

Vedere la sezione Array del manuale per una spiegazione dettagliata di come gli array siano implementati ed usati in PHP.

Requisiti

Queste funzioni sono disponibili nei moduli standard, che sono sempre disponibili.

Istallazione

Non ‘ necessaria nessuna installazione per usare queste funzioni, esse fanno parte del core di PHP.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

Questa estensione non definisce alcun tipo di risorsa.

Costanti Predefinite

`CASE_UPPER` e `CASE_LOWER` sono usate con la funzione array_change_key_case(). Sono usate rispettivamente per cambiare una stringa in caratteri maiuscoli o minuscoli.

Vedere anche

Vedere anche is_array(), explode(), implode(), split() e join().

array (unknown)

Crea un array

array **array** ([mixed ...]) \linebreak

Restituisce un array contenente i parametri. Ai parametri si può dare un indice con l'operatore =>.

> **Nota: array()** è un costrutto del linguaggio usato per rappresentare array letterali, e non una normale funzione.

La sintassi "indice => valori", separati da virgole, definisce indici e valori. indice può essere di tipo string o numerico. Quando l'indice è omesso, viene generato automaticamente un indice intero, a partire da 0. Se l'indice è un intero, il successivo indice generato sarà l'indice intero più grande + 1. Si noti che quando due indici identici vengono definiti, l'ultimo sovrascrive il primo.

L'esempio seguente dimostra come creare un array bidimensionale, come specificare le chiavi per gli array associativi, e come modificare la serie degli indici numerici negli array normali.

Esempio 1. Esempio di array()

```
$frutta = array (
    "frutta"  => array ("a"=>"arancia", "b"=>"banana", "c"=>"mela"),
    "numeri" => array (1, 2, 3, 4, 5, 6),
    "buche"   => array ("prima", 5 => "seconda", "terza")
);
```

Esempio 2. Indice automatico con array()

```
$array = array( 1, 1, 1, 1,  1, 8=>1,  4=>1, 19, 3=>13);
print_r($array);
```

che stamperà:

```
Array
(
    [0] => 1
    [1] => 1
    [2] => 1
    [3] => 13
    [4] => 1
    [8] => 1
    [9] => 19
)
```

Si noti che l'indice '3' è definito due volte, e che mantiene il valore finale 13. L'indice 4 è definito dopo l'indice 8, e il successivo indice generato (valore 19) è 9, dal momento che l'indice più grande era 8.

Questo esempio crea un array che parte da 1 (1-based).

Esempio 3. Indice 1-based con array()

```
        $primotrimestre  = array(1 => 'Gennaio', 'Febbraio', 'Marzo');
        print_r($primotrimestre);
```

che stamperà:

```
Array
(
    [1] => 'Gennaio'
    [2] => 'Febbraio'
    [3] => 'Marzo'
)
```

Vedere anche array_pad(), list() e range()

array_change_key_case (PHP 4 >= 4.2.0)

Restituisce un array con tutte le chiavi cambiate in maiuscolo o in minuscolo

array **array_change_key_case** (array input [, int case]) \linebreak

array_change_key_case() cambia le chiavi nell'array `input` in modo che siano tutte minuscole o maiuscole. Il tipo di cambiamento dipende dal parametro opzionale `case`. Si possono usare due costanti, `CASE_UPPER` per le maiuscole e `CASE_LOWER` per le minuscole. Il default è `CASE_LOWER`. La funzione non modifica le chiavi numeriche.

Esempio 1. esempio di array_change_key_case()

```
$input_array = array("PriMo" => 1, "SecOndO" => 4);
print_r(array_change_key_case($input_array, CASE_UPPER));
```

Il risultato di questo programma sarà:

```
Array
```

```
(
    [PRIMO] => 1
    [SECONDO] => 2
)
```

array_chunk (PHP 4 >= 4.2.0)

Spezza un array in tronconi

array **array_chunk** (array input, int dimensione [, bool mantieni_chiavi]) \linebreak

array_chunk() spezza l'array in più array di dimensione *dimensione*. L'ultimo array potrebbe ovviamente avere una dimensione inferiore. Gli array sono restituiti in un array multidimensionale indicizzato con chiavi che partono da zero.

Impostando il parametro opzionale *preserve_keys* a TRUE, si forza PHP a mantenere le chiavi originarie dell'array di input. Se si imposta a FALSE come chiavi verranno usati in ogni array dei numeri crescenti a partire da zero. Il default è FALSE.

Esempio 1. esempio di array_chunk()

```
$input_array = array('a', 'b', 'c', 'd', 'e');
print_r(array_chunk($input_array, 2));
print_r(array_chunk($input_array, 2, TRUE));
```

Il risultato di questo programma sarà:

```
Array
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [0] => c
            [1] => d
        )

    [2] => Array
        (
            [0] => e
        )

)
Array
```

```
(
    [0] => Array
        (
            [0] => a
            [1] => b
        )

    [1] => Array
        (
            [2] => c
            [3] => d
        )

    [2] => Array
        (
            [4] => e
        )

)
```

array_count_values (PHP 4 >= 4.0.0)

Conta tutti i valori di un array

array **array_count_values** (array input) \linebreak

array_count_values() restituisce un array che ha i valori dell'array *input* per chiavi e la loro frequenza in *input* come valori.

Esempio 1. Esempio di array_count_values()

```
$array = array (1, "ciao", 1, "mondo", "ciao");
print_r(array_count_values ($array));
```

Il risultato di questo programma sarà:

```
Array
(
    [1] => 2
    [ciao] => 2
    [mondo[ => 1
)
```

array_diff (PHP 4)

Calcola la differenza di due o più array

array **array_diff** (array array1, array array2 [, array ...]) \linebreak

array_diff() restituisce un array contenente tutti i valori di *array1* che non sono presenti in alcuno degli altri array. Si noti che le associazioni con le chiavi vengono mantenute.

Esempio 1. Esempio di array_diff()

```
$array1 = array ("a" => "verde", "rosso", "blu", "rosso");
$array2 = array ("b" => "verde", "giallo", "rosso");
$risultato = array_diff ($array1, $array2);
```

In questo modo `$risultato` sarà `array ("blue");`. Occorrenze multiple in $array1 sono tutte trattate nello stesso modo.

> **Nota:** Due elementi sono considerati uguali se e solo se `(string) $elem1 === (string) $elem2`. Ovvero: quando la rappresentazione sotto forma di stringa è la stessa.

> **Attenzione**
>
> Questa funzione era errata nel PHP 4.0.4!

Vedere anche array_intersect().

array_fill (PHP 4 >= 4.2.0)

Riempie un array con i valori specificati

array **array_fill** (int inizio, int num, mixed valore) \linebreak

array_fill() riempie un array con *num* elementi inizializzati con il valore del parametro *valore*, e con le chiavi che partono dal valore del parametro *start_index*.

Esempio 1. esempio di array_fill()

```
$a = array_fill(5, 6, ’banana’);
```

$a ora ha i seguenti elementi (usando la funzione print_r()):

```
Array
```

```
(
    [5]  => banana
    [6]  => banana
    [7]  => banana
    [8]  => banana
    [9]  => banana
    [10] => banana
)
```

array_filter (PHP 4 >= 4.0.6)

Filtra gli elementi di un array usando una funzione callback

array **array_filter** (array input [, mixed callback]) \linebreak

array_filter() restituisce un array contenente tutti gli elementi di `input` filtrati attraverso una funzione callback. Se `input` è un array associativo le chiavi sono mantenute.

Esempio 1. Esempio di array_filter()

```
function dispari($var) {
    return ($var % 2 == 1);
}

function pari($var) {
    return ($var % 2 == 0);
}

$array1 = array ("a"=>1, "b"=>2, "c"=>3, "d"=>4, "e"=>5);
$array2 = array (6, 7, 8, 9, 10, 11, 12);

echo "Dispari :\n";
print_r(array_filter($array1, "dispari"));
echo "Pari :\n";
print_r(array_filter($array2, "pari"));
```

Il risultato di questo sarà:

```
Dispari :
Array
(
    [a] => 1
    [c] => 3
    [e] => 5
)
Pari:
```

```
Array
(
    [0] => 6
    [2] => 8
    [4] => 10
    [6] => 12
)
```

> **Nota:** Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

Gli utenti non possono modificare l'array attraverso la funzione di callback, ad esempio aggiungere/togliere un elemento, o cancellare l'array su cui **array_filter()** è applicata. Se l'array viene cambiato, il comportamento di questa funzione non è definito.

Vedere anche array_map() e array_reduce().

array_flip (PHP 4 >= 4.0.0)

Scambia tutti i valori di un array

array **array_flip** (array trans) \linebreak

array_flip() restituisce un array scambiato, ovvero le chiavi di `trans` diventano valori e i valori di `trans` diventano chiavi.

Si noti che i valori di `trans` devono poter diventare chiavi valide, ovvero devo essere di tipo integer o string. Un errore verrà segnalato se un valore ha il tipo errato, e la coppia chiave/valore in questione *non verrà scambiata*.

Se un valore ha più di una occorrenza, L'ultima chiave verrà usata come valore, e tutte le altre verranno perse.

array_flip() restituisce `FALSE` se fallisce.

Esempio 1. Esempio di array_flip()

```
$trans = array_flip ($trans);
$original = strtr ($str, $trans);
```

Esempio 2. Esempio di array_flip(): collisione

```
$trans = array ("a" => 1, "b" => 1, "c" => 2);
$trans = array_flip ($trans);
print_r($trans);
```

ora $trans è:

```
Array
(
    [1] => b
    [2] => c
)
```

array_intersect (PHP 4)

Calcola l'intersezione degli arrays

array **array_intersect** (array array1, array array2 [, array ...]) \linebreak

array_intersect() restituisce un array contenente tutti i valori di *`array1`* che siano presenti in tutti gli array passati come argomento. Si noti che le associazioni con le chiavi sono mantenute.

Esempio 1. Esempio di array_intersect()

```
$array1 = array ("a" => "verde", "rosso", "blu");
$array2 = array ("b" => "verde", "giallo", "rosso");
$risultato = array_intersect ($array1, $array2);
```

In questo modo `$result` sarà:

```
Array
(
    [a] => verde
    [0] => rosso
)
```

Nota: Due elementi sono considerati uguali solo e solo se `(string) $elem1 === (string) $elem2`. Ovvero: quando la rappresentazione sotto forma di stringa è la stessa.

Attenzione

Questa funzione era errata nel PHP 4.0.4!

Vedere anche array_diff().

array_key_exists (PHP 4 >= 4.1.0)

Controlla se l'indice (o chiave) specificato esiste nell'array

bool **array_key_exists** (mixed chiave, array cerca) \linebreak

array_key_exists() restituisce `TRUE` se il parametro *`chiave`* esiste nell'array. *`chiave`* può essere qualsiasi valore accettabile per un indice di array.

Esempio 1. esempio di array_key_exists()

```
$un_array = array("primo" => 1, "secondo" => 4);
if (array_key_exists("primo", $un_array)) {
    echo "L'elemento 'primo' è nell'array";
}
```

Nota: Il nome di questa funzione è **key_exists()** nel PHP versione 4.0.6.

Vedere anche isset().

array_keys (PHP 4 >= 4.0.0)

Restituisce tutte le chiavi di un array

array **array_keys** (array input [, mixed search_value]) \linebreak

array_keys() rstituisce le chiavi, numeriche e stringa, dell'array *`input`*.

Se il parametro opzionale *`search_value`* è specificato, solo le chiavi con che corrispondono a quel valore vengono restituite. Altrimenti, vengono restituite tutte le chiavi dell'array *`input`*.

Esempio 1. Esempio di array_keys()

```
$array = array (0 => 100, "colore" => "rosso");
print_r(array_keys ($array))

$array = array ("blu", "rosso", "verde", "blu", "blu");
print_r(array_keys ($array, "blu"));

$array = array ("colore" => array("blu", "rosso", "verde"), "misura" =&gt; ar-
ray("piccola", "media", "grande"));
print_r(array_keys ($array));
```

Il risultato di questo programma sarà:

```
Array
(
    [0] => 0
    [1] => colore
)
Array
(
    [0] => 0
    [1] => 3
    [2] => 4
)
Array
(
    [0] => colore
    [1] => misura
)
```

Nota: Questa funzione è stata aggiunta in PHP 4, qui sotto c'è una implementazione per coloro che usano ancora PHP 3.

Esempio 2. Implementazione di array_keys() per utenti PHP 3

```
function array_keys ($arr, $term="") {
    $t = array();
    while (list($k,$v) = each($arr)) {
        if ($term && $v != $term) {
            continue;
        }
        $t[] = $k;
    }
    return $t;
}
```

Vedere anche array_values().

array_map (PHP 4 >= 4.0.6)

Applica la funzione callback a tutti gli elementi dell'array dato

array **array_map** (mixed callback, array arr1 [, array arr2...]) \linebreak

array_map() restituisce un array contenente tutti gli elementi di *arr1* dopo che è stata loro applicata la funzione callback. Il numero di parametri che la funzione callback accetta deve corrispondere al numero di array passati alla funzione **array_map()**

Esempio 1. Esempio di array_map()

```
function cubo($n) {
    return $n*$n*$n;
}

$a = array(1, 2, 3, 4, 5);
$b = array_map("cubo", $a);
print_r($b);
```

In questo modo `$b` sarà:

```
Array
(
    [0] => 1
    [1] => 8
    [2] => 27
    [3] => 64
    [4] => 125
)
```

Esempio 2. array_map() - usare più array

```
function mostra_Spagnolo($n, $m) {
   return "Il numero $n si dice $m in Spagnolo";
}

function mappa_Spagnolo($n, $m) {
    return array ($n => $m);
```

```
}

$a = array(1, 2, 3, 4, 5);
$b = array("uno", "dos", "tres", "cuatro", "cinco");

$c = array_map("mostra_Spagnolo", $a, $b);
print_r($c);

$d = array_map("mappa_Spagnolo", $a, $b);
print_r($d);
```

Questo restituisce:

```
//stampa di $c
Array
(
    [0] => Il numero 1 si dice uno in Spagnolo
    [1] => Il numero 2 si dice dos in Spagnolo
    [2] => Il numero 3 si dice tres in Spagnolo
    [3] => Il numero 4 si dice cuatro in Spagnolo
    [4] => Il numero 5 si dice cinco in Spagnolo
)

// stampa di $d
Array
(
    [0] => Array
        (
            [1] => uno
        )

    [1] => Array
        (
            [2] => dos
        )

    [2] => Array
        (
            [3] => tres
        )

    [3] => Array
        (
            [4] => cuatro
        )

    [4] => Array
        (
            [5] => cinco
        )

)
```

Generalmente, quando si usano due o più array, questi devono avere eguale lunghezza in quanto la funzione callback viene applicata in parallelo agli elementi corrispondenti. Se gli array sono di lunghezza diversa, il più corto verrà esteso con elementi vuoti.

Un uso interessante di questa funzione è quello di costruire un array di array, cosa che pouò essere facilmente ottenuta usando `null` come nome della funzione callback

Esempio 3. Array_map() - creare un array di array

```
$a = array(1, 2, 3, 4, 5);
$b = array("uno", "due", "tre", "quattro", "cinque");
$c = array("uno", "dos", "tres", "cuatro", "cinco");

$d = array_map(null, $a, $b, $c);
print_r($d);
```

Il risultato di questo programma sarà;

```
Array
(
    [0] => Array
        (
            [0] => 1
            [1] => uno
            [2] => uno
        )

    [1] => Array
        (
            [0] => 2
            [1] => due
            [2] => dos
        )

    [2] => Array
        (
            [0] => 3
            [1] => tre
            [2] => tres
        )

    [3] => Array
        (
            [0] => 4
            [1] => quattro
            [2] => cuatro
        )

    [4] => Array
        (
```

```
                [0] => 5
                [1] => cinque
                [2] => cinco
            )

    )
```

Vedere anche array_filter() e array_reduce().

array_merge (PHP 4 >= 4.0.0)

Fonde due o più array

array **array_merge** (array array1, array array2 [, array ...]) \linebreak

array_merge() fonde gli elementi di due o più array in modo che i valori di un array siano accodati a quelli dell'array precedente. Restituisce l'array risultante.

Se gli array in input hanno le stesse chiavi stringa, l'ultimo valore di quella chiave sovrascriverà i precedenti. Comunque, se gli array hanno le stesse chiavi numeriche, l'ultimo valore **non** sovrascriverà quello originale, bensì sarà accodato.

Esempio 1. Esempio di array_merge()

```
$array1 = array ("colore" => "rosso", 2, 4);
$array2 = array ("a", "b", "colore" => "verde", "forma" => "trapezio", 4);
$risultato = array_merge ($array1, $array2);
```

La variabile `$risultato` sarà:

```
Array
(
    [colore] => verde
    [0] => 2
    [1] => 4
    [2] => a
    [3] => b
    [forma] => trapezio
    [4] => 4
)
```

Esempio 2. Esempio di array_merge()

```
$array1 = array();
$array2 = array(1 => "dati");
$result = array_merge($array1, $array2);
```

Non dimenticarsi che le chiavi numeriche saranno rinumerate!

```
Array
(
    [0] => data
)
```

Se si vogliono preservare gli array e li si vuole solo concatenare, usare l'operatore +:

```
$array1 = array();
$array2 = array(1 => "dati");
$result = $array1 + $array2;
```

La chiave numerica sarà preservata e così pure l'associazione.

```
Array
(
    [1] => data
)
```

Vedere anche array_merge_recursive().

array_merge_recursive (PHP 4)

Fonde due o più array in modo ricorsivo

array **array_merge_recursive** (array array1, array array2 [, array ...]) \linebreak

Array_merge_recursive() fonde gli elementi di due o più array in modo tale che i valori di un array siano accodati all'array precedente. Restituisce l'array risultante.

Se gli array in input hanno le stesse chiavi stringa, i valori di queste chiavi vengono fusi in un array, e questo è fatto in modo ricorsivo, cio' se uno dei valori è un array, la funzione lo fonder%agrave; con una voce corrispondente in un altro array Comunque, se gli array hanno la stessa chiave numerica, l'ultimo valore non sovrascriver' il valore originale, bensì verrà accodato.

Esempio 1. Esempio di array_merge_recursive()

```
$ar1 = array ("colore" => array ("preferito" => "rosso"), 5);
$ar2 = array (10, "colore" => array ("preferito" => "verde", "blu"));
$risultato = array_merge_recursive ($ar1, $ar2);
```

La variabile `$risultato` sarà:

```
Array
(
    [colore] => Array
        (
             [preferito] => Array
                (
                    [0] => rosso
                    [1] => verde
                )

            [0] => blu
        )

    [0] => 5
    [1] => 10
)
```

Vedere anche array_merge().

array_multisort (PHP 4 >= 4.0.0)

Ordina array multipli o multidimensionali

bool **array_multisort** (array ar1 [, mixed arg [, mixed ... [, array ...]]]) \linebreak

Array_multisort() Può essere usata per ordinare parecchi array allo stesso tempo, oppure un array multidimensionale, rispetto a una o più dimensioni. Mantiene le associazioni delle chiavi durante l'ordinamento.

Gli array in input sono trattati come campi di una tabella che vengano ordinati per righe - questo assomiglia alla funzionalità della clausola SQL ORDER BY Il primo array è quello primario, rispetto a cui aordinare. Le righe (valori) in questo array that siano uguali vengono ordinate secondo l'array successivo, e così via.

La struttura degli argomenti di questa funzione è un po' inusuale, ma flessibile. Il primo argomento deve essere un array. In seguito, ogni argomento può essere sia un aray che un flag di ordinamento, selezionabile dalla seguente lista.

Flag di ordinamento:

- SORT_ASC - ordinamento crescente
- SORT_DESC - ordinamento decrescente

Sorting type flags:

- SORT_REGULAR - confronta gli elementi in modo normale
- SORT_NUMERIC - confronta gli elementi numericamente
- SORT_STRING - confronta gli elementi come stringhe

Dopo ogni array, non si possono specificare due flag dello stesso tipo. I flag specificati dopo un array si applicano solo a quell'array - sono reimpostati ai default SORT_ASC e SORT_REGULAR dopo ogni nuovo array passato come argomento.

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

Esempio 1. Ordinamre più array

```
$ar1 = array ("10", 100, 100, "a");
$ar2 = array (1, 3, "2", 1);
array_multisort ($ar1, $ar2);
```

In questo esempio, dopo l'ordinamento, il primo array conterrà "10", "a", 100, 100. Il secondo array conterrà 1, 1, "2", 3. Gli elementi nel secondo array che corrispondono agli elementi identici nel primo array (100 e 100) vengono pure ordinati.

Esempio 2. Ordinare un array multi-dimensionale

```
$ar = array (array ("10", 100, 100, "a"), array (1, 3, "2", 1));
array_multisort ($ar[0], SORT_ASC, SORT_STRING,
                 $ar[1], SORT_NUMERIC, SORT_DESC);
```

In questo esempio, dopo l'ordinamento, il primo array conterrà 10, 100, 100, "a" (ordinato come stringhe ordine crescente), e il secondo conterrà 1, 3, "2", 1 (ordinati come numeri, in ordine decrescente).

array_pad (PHP 4 >= 4.0.0)

Riempie con un valore un array fino alla lunghezza specificata

array **array_pad** (array input, int pad_size, mixed pad_value) \linebreak

array_pad() restituisce una copia di *`input`* allungato alla dimensione sepcificata da *`pad_size`* con il valore *`pad_value`*. Se *`pad_size`* è positivo l'array è riempito sulla destra, se è negativo sulla sinistra. Se il valore assoluto di *`pad_size`* è minore o uguale alla lunghezza di *`input`* non viene effettuata alcuna modifica.

Esempio 1. esempio di array_pad()

```
$input = array (12, 10, 9);

$risultato = array_pad ($input, 5, 0);
// risultato diventa array (12, 10, 9, 0, 0)

$risultato = array_pad ($input, -7, -1);
// risultato diventa array (-1, -1, -1, -1, 12, 10, 9)

$risultato = array_pad ($input, 2, "noop");
// ridimensionamento non efettuato
```

array_pop (PHP 4 >= 4.0.0)

Estrae l'elemento alla fine dell'array

mixed **array_pop** (array array) \linebreak

array_pop() estrae e restituisce l'ultimo valore di *`array`*, accorciando *`array`* di un elemento. Se *`array`* è vuoto (o non è un array), viene restituito `NULL`.

Esempio 1. esempio di array_pop()

```
$pila = array ("arancia", "banana", "mela", "lampone");
$frutto = array_pop ($pila);
```

Dopo questa istruzione, `$pila` avrà solo 3 elementi:

```
Array
(
    [0] => arancia
    [1] => banana
    [2] => mela
)
```

e `lampone` verràa assegnato alla variabile `$frutto`.

> **Attenzione**
>
> Questa funzione può restituire il Booleano `FALSE`, ma può anche restituire un valore non-Booleano valutato come `FALSE`, come ad esempio `0` o "". Per favore fare riferimento alla sezione Booleans per maggiorni informazioni. Usare l'operatore === per controllare il valore restituito da questa funzione.

Vedere anche array_push(), array_shift() e array_unshift().

array_push (PHP 4 >= 4.0.0)

Accoda uno o più elementi ad un array

int **array_push** (array array, mixed var [, mixed ...]) \linebreak

array_push() tratta *`array`* come una pila, e accoda le variabili date alla fine di *`array`*. La lunghezza di *`array`* aumenta del numero di variabili accodate. Ha lo stesso effetto di:

```
$array[] = $var;
```

ripetuto per ogni *`var`*.

Restituisce il nuovo numero di elementi nell'array.

Esempio 1. esempio di array_push()

```
$pila = array ("arancia", "banana");
array_push ($pila, "mela", "lampone");
```

In questo esempio `$pila` avrà i seguenti elementi:

```
Array
(
    [0] => arancia
    [1] => banana
    [2] => mela
    [3] => lampone
)
```

Vedere anche array_pop(), array_shift() e array_unshift().

array_rand (PHP 4 >= 4.0.0)

Estrae a caso uno o più elementi da un array

mixed **array_rand** (array input [, int num_req]) \linebreak

array_rand() è piuttosto utile quando si vuole estrarre a caso uno o più elementi da un array. Prende un array (*input*) e un argomento ozpionale (*num_req*) che specifica quanti elementi estrarre - se non è specificato, è 1 per default.

Se si sta estraendo solo un elemento, **array_rand()** restituisce la chiave di un elemento. Altrimenti, restituisce un array di chiavi. Questo viene fatto in modo da permettere di estrarre dall'array sia le chiavi che i valori.

Non dimenticare di chiamare srand() per perturbare il generatore di numri casuali.

Esempio 1. esempio di array_rand()

```
srand ((float) microtime() * 10000000);
$input = array ("Neo", "Morpheus", "Trinity", "Cypher", "Tank");
$chiavi = array_rand ($input, 2);
print $input[$chiavi[0]]."\n";
print $input[$chiavi[1]]."\n";
```

array_reduce (PHP 4 >= 4.0.5)

Riduce iterativamente l'array a un singolo valore utilizzando una funzione callback

mixed **array_reduce** (array input, mixed callback [, int initial]) \linebreak

array_reduce() applica iterativamente la funzione *callback* agli elementi dell'array *input*, riducendo l'array a un singolo valore. Seil parametro opzionale *intial* è specificato, viene usato come valore iniziale all'inizio del processo, o come risultato finale nel caso l'array sia vuoto.

Esempio 1. esempio di array_reduce()

```
function rsum($v, $w) {
    $v += $w;
    return $v;
}

function rmul($v, $w) {
    $v *= $w;
    return $v;
}

$a = array(1, 2, 3, 4, 5);
```

```
$x = array();
$b = array_reduce($a, "rsum");
$c = array_reduce($a, "rmul", 10);
$d = array_reduce($x, "rsum", 1);
```

In questo modo `$b` conterrà `15`, `$c` conterrà `1200` (= 1*2*3*4*5*10) e `$d` conterrà `1`.

Vedere anche array_filter() e array_map().

array_reverse (PHP 4 >= 4.0.0)

Restituisce un array con gli elementi in ordine invertito

array **array_reverse** (array array [, bool mantieni_chiavi]) \linebreak

array_reverse() prende *`array`* e restituisce un nuovo array con l'ordine degli elementi invertito, mantenendo le chiavi sie *`mantieni_chiavi`* è `TRUE`.

Esempio 1. esempio di array_reverse()

```
$input = array ("php", 4.0, array ("verde", "rosso"));
$risultato = array_reverse ($input);
$resultato_chiavi = array_reverse ($input, TRUE);
```

Questo fa sì che sia `$risultato` che `$risultato_chiavi` abbiano gli stessi elementi, ma si noti la differenza tra le chiavi. La stampa di `$risultato` e `$risultato_chiavi` sarà:

```
Array
(
    [0] => Array
        (
            [0] => verde
            [1] => rosso
        )

    [1] => 4
    [2] => php
)
Array
(
    [2] => Array
        (
            [0] => verde
            [1] => rosso
        )

    [1] => 4
    [0] => php
```

)

Nota: Il secondo parametro è stato aggiunto in PHP 4.0.3.

array_search (PHP 4 >= 4.0.5)

Ricerca un dato valore in un array e ne restituisce la chiave corrispondente, se la ricerca ha successo.

mixed **array_search** (mixed ago, array pagliaio [, bool strict]) \linebreak

Cerca in *pagliaio* per trovare *ago* e restituisce la chiave se viene trovato nell'array, `FALSE` altrimenti.

> **Nota:** Nelle versioni di PHP antecedenti la 4.2.0, **array_search()** restituisce `NULL` invece di `FALSE` in caso di fallimento.

Se il terzo parametro opzionale *strict* è impostato a `TRUE` la funzione **array_search()** controllerè anche il tipo di *ago* nell'array *pagliaio*.

> **Attenzione**
>
> Questa funzione può restituire il Booleano `FALSE`, ma può anche restituire un valore non-Booleano valutato come `FALSE`, come ad esempio `0` o "". Per favore fare riferimento alla sezione Booleans per maggiorni informazioni. Usare l'operatore === per controllare il valore restituito da questa funzione.

Vedere anche in_array().

array_shift (PHP 4 >= 4.0.0)

Estrae l'elemento alla testa dell'array

mixed **array_shift** (array array) \linebreak

array_shift() estrae il primo elemento di *array* e lo restituisce, accorciando *array* di un elemento e spostando tutti gli altri all'indietro. Se *array* è vuoto (o non è un array), viene restituito `NULL`.

Esempio 1. esempio di array_shift()

```
$pila = array ("arancia", "banana", "mela", lampone");
$frutto = array_shift ($pila);
```

In questo modo `$pila` rimarrà con 3 elementi:

```
Array
(
    [0] => banana
    [1] => mela
    [2] => lampone
)
```

e `arancia` sarà assegnata a `$frutto`.

Vedere anche array_unshift(), array_push() e array_pop().

array_slice (PHP 4 >= 4.0.0)

Estrae un sottoinsieme da un array

array **array_slice** (array array, int offset [, int length]) \linebreak

array_slice() restituisce la sequenza di elementi dell'array *`array`* come specificato dai parametri *`offset`* e *`length`* .

Se *`offset`* è positivo, la sequenza comincerà da quell'offset in *`array`*. Se *`offset`* è negativo, la sequenza comincerà alla distanza *`offset`* dalla fine di *`array`*.

Se *`length`* è specificata ed è positiva, la sequenza conterrà quel numero di elementi. Se *`length`* è specificata ed è negativa la sequenza si fermerà a quel numero di elementi dalla fine dell'array. Se viene omessa, la sequenza conterrà tutto da *`offset`* fino alla fine di *`array`*.

Si noti che **array_slice()** ignorerà le chiavi dell'array, e calcolerè gli spiazzamenti e le lunghezze basandosi sulle posizioni correnti degli elementi nell'array.

Esempio 1. esempi di array_slice()

```
$input = array ("a", "b", "c", "d", "e");

$output = array_slice ($input, 2);      // restituisce "c", "d" e "e"
$output = array_slice ($input, 2, -1);  // restituisce "c", "d"
$output = array_slice ($input, -2, 1);  // restituisce "d"
$output = array_slice ($input, 0, 3);   // restituisce "a", "b" e "c"
```

Vedere anche array_splice().

array_splice (PHP 4 >= 4.0.0)

Rimuove una porzione dell'array e la sostituisce con altro

array **array_splice** (array input, int offset [, int length [, array replacement]]) \linebreak

array_splice() rimuove gli elementi specificati da *offset* e *length* dall'array *input*, e li sostituisce con gli elementi dell'array *replacement*, se fornito. Restituisce un array contenente gli elementi estratti.

Se *offset* è positivo l'inizio della porzione rimossa è a quella distanza dall'inizio dell'array *input*. Se *offset* è negativo inizia a quella distanza dalla fine dell'array *input*.

Se *length* è omessa, rimuove tutti gli elementi da *offset* alla fine dell'array. Se *length* è specificata a positiva, quel numero di elementi vengono rimossi. Se *length* è specificata e negativa la porzione da rimuovere terminerà a *length* elementi dalla fine dell'array. Suggerimento: per rimuovere tutti gli elementi tra *offset* e la fine dell'array quando è specificato pure *replacement*, usare `count($input)` nel parametro *length*.

Se l'array *replacement* è specificato, gli elementi rimossi sono sostituiti dagli elementi di questo array. Se *offset* e *length* sono tali per cui niente viene rimosso, gli elementi dell'array *replacement* sono inseriti nella posizione specificata da *offset*. Suggerimento: se *replacement* è composto solo da un elemento non è necessario porlo nel costrutto `array()`, a meno che l'elemento stesso non sia un array.

Valgono le seguenti equivalenze:

```
array_push ($input, $x, $y)     array_splice ($input, count ($input), 0,
                                              array ($x, $y))
array_pop ($input)              array_splice ($input, -1)
array_shift ($input)            array_splice ($input, 0, 1)
array_unshift ($input, $x, $y)  array_splice ($input, 0, 0, array ($x, $y))
$a[$x] = $y                     array_splice ($input, $x, 1, $y)
```

Restituisce un array contenente gli elementi rimossi.

Esempio 1. esempi di array_splice()

```
$input = array ("rosso", "verde", "blu", "giallo");
array_splice ($input, 2);
// $input è ora array ("rosso", "verde")

$input = array ("rosso", "verde", "blu", "giallo");
array_splice ($input, 1, -1);
// $input è ora array ("rosso", "giallo")
```

```
$input = array ("rosso", "verde", "blu", "giallo");
array_splice ($input, 1, count($input), "arancio");
// $input è ora array ("rosso", "arancio")

$input = array ("rosso", "verde", "blu", "giallo");
array_splice ($input, -1, 1, array("nero", "marrone"));
// $input è ora array ("rosso", "verde",
//          "blu", "nero", "marrone")
```

Vedere anche array_slice().

array_sum (PHP 4 >= 4.0.4)

Calcola la somma dei valori di un array.

mixed **array_sum** (array array) \linebreak

array_sum() restituisce la somma dei valori dell’array sotto forma di integer o float.

Esempio 1. esempi di array_sum()

```
$a = array(2, 4, 6, 8);
echo "sum(a) = ".array_sum($a)."\n";

$b = array("a"=>1.2,"b"=>2.3,"c"=>3.4);
echo "sum(b) = ".array_sum($b)."\n";
```

Il risultato di questo programma sarà:

```
sum(a) = 20
sum(b) = 6.9
```

Nota: Le versioni di PHP antecedenti alla 4.0.6 modificavano l’array stesso e convertivano le stringhe in numeri (le quali erano convertite in zeri la maggior parte delle volte, a seconda dal valore).

array_unique (PHP 4)

Rimuove i valori duplicati di un array

array **array_unique** (array array) \linebreak

array_unique() prende *array* e restituisce un nuovo array senza i valori duplicati.

Si noti che le chiavi sono mantenute. **array_unique()** ordina i valori trattandoli come stringhe, quindi mantiene la prima chiave trovata per ogni valore, e ignorerà tutte le altre chiavi. Questo non significa che la chiave del primo valore dell'*array* non ancora ordinato verrà mantenuta.

> **Nota:** Due elementi sono considerati uguali se e solo se `(string) $elem1 === (string) $elem2`. Ovvero: quando la rappresentazione sotto forma di stringa è la stessa.
>
> Verrà usato il primo elemento.

Attenzione

Questa funzione era errata nel PHP 4.0.4!

Esempio 1. esempio di array_unique()

```
$input = array ("a" => "verde", "rosso", "b" => "verde", "blu", "rosso");
$risultato = array_unique ($input);
print_r($result);
```

Questo mostrerà:

```
Array
(
    [b] => verde
    [1] => blu
    [2] => rosso
)
```

Esempio 2. array_unique() e i tipi

```
$input = array (4,"4","3",4,3,"3");
$risultato = array_unique ($input);
var_dump($risultato);
```

Il risultato di questo programma sarà (PHP 4.0.6):

```
array(2) {
  [0]=>
  int(4)
  [2]=>
  string(1) "3"
}
```

array_unshift (PHP 4 >= 4.0.0)

Inserisce uno o più elementi all'inizio dell'array

int **array_unshift** (array array, mixed var [, mixed ...]) \linebreak

array_unshift() aggiunge gli elementi specificati in testa ad *array*. Si noti che la lista di elementi è aggiunta in blocco, in modo tale che gli elementi rimangano nello stesso ordine.

Restituisce il nuovo numero di elementi in *array*.

Esempio 1. esempio di array_unshift()

```
$lista = array ("arancia", "banana");
array_unshift ($lista, "mela", "lampone");
```

In questo modo `$lista` conterrà i seguenti elementi:

```
Array
(
    [0] => mela
    [1] => lampone
    [2] => arancia
    [3] => banana
)
```

Vedere anche array_shift(), array_push() e array_pop().

array_values (PHP 4 >= 4.0.0)

Restituisce tutti i valori di un array

array **array_values** (array input) \linebreak

array_values() restituisce tutti i valori dell'array *input*.

Esempio 1. esempio di array_values()

```
$array = array ("taglia" => "XL", "colore" => "oro");
print_r(array_values ($array));
```

Questo mostrerà:

```
Array
(
    [0] => XL
    [1] => oro
)
```

Nota: Questa funzione è stata aggiunta in PHP 4, qui sotto si trova una implementazione per chi usa ancora PHP 3.

Esempio 2. Implementazone di array_values() per gli utenti PHP 3

```
function array_values ($arr) {
    $t = array();
    while (list($k, $v) = each ($arr)) {
        $t[] = $v;
    }
    return $t;
}
```

Vedere anche array_keys().

array_walk (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Esegue una funzione su ogni elemento dell'array

int **array_walk** (array array, string funzione [, mixed datiutente]) \linebreak

Esegue la funzione definita dall'utente identificata da `funzione` su ogni elemento di `array`. A `funzione` verrà passato il valore dell'elemento come primo parametro e la chiave come secondo parametro. Se `datiutente` è specificato, verrà passato come terzo parametro alla funzione. `funzione` deve essere una funzione definita dall'utente, e non può essere una funzione nativa PHP. Quindi, non si può usare **array_walk()** direttamente con **str2lower()**, bensì occorre costruire una funzione utente con tale istruzione, e passarla come argomento.

> **Nota:** Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

Se `funzione` richiede più di due o tre argomenti, a seconda di `datiutente`, un warning verrà generato ogni qualvolta **array_walk()** chiami `funzione`. Questi warning possono essere soppressi apponendo il simbolo '@' alla chiamata di **array_walk()**, oppure usando error_reporting().

> **Nota:** Se `funzione` deve lavorare con i reali valori dell'array, specificare che il primo parametro di `funzione` deve essere passato come riferimento. A qesto punto ogni modifica a questi elementi verrà effettuata sull'array stesso.
>
> Modificare l'array all'interno di `func` può causare comportamenti imprevedibili.

> **Nota:** Il passaggio della chiave e di `datiutente` a `func` è stato aggiunto nella versione 4.0.
>
> In PHP 4 la funzione reset() deve essere chiamata obbligatoriamente, in quanto **array_walk()** non reinizializza automaticamente l'array.
>
> Gli utenti non possono modificare l'array attraverso la funzione di callback, ad esempio aggiungere/togliere un elemento, o cancellare l'array su cui **array_walk()** è applicata. Se l'array viene cambiato, il comportamento di questa funzione non è definito.

Esempio 1. esempio di array_walk()

```
$frutta = array ("d"=>"limone", "a"=>"arancia", "b"=>"banana", "c"=>"mela");

function modifica (&$elemento1, $chiave, $prefisso) {
    $elemento1 = "$prefisso: $elemento1";
}

function stampa ($elemento2, $chiave) {
    echo "$chiave. $elemento2<br>\n";
}
echo "Prima ...:\n";
array_walk ($frutta, 'stampa');
reset ($frutta);
array_walk ($frutta, 'modifica', 'frutto');
```

```
echo "... e dopo:\n";
reset ($frutta);
array_walk ($frutta, 'stampa');
```

Il risultato del programma sarà:

```
Prima ...:
d. limone
a. arancia
b. banana
c. mela
... and after:
d. frutto: limone
a. frutto: arancia
b. frutto: banana
c. frutto: mela
```

Vedere anche each() e list().

arsort (PHP 3, PHP 4 >= 4.0.0)

Ordina un array in ordine decrescente e mantiene le associazioni degli indici

void **arsort** (array array [, int sort_flags]) \linebreak

Questa funzione ordina un array in modo tale che i suoi indici mantengano la loro correlazione con gli elementi ai quali sono associati. Viene usata principalmente nell'ordinamento degli array associativi, quando la disposizione originaria degli elementi è importante.

Esempio 1. esempio di arsort()

```
$frutta = array ("d"=>"limone", "a"=>;"arancia", "b"=>"banana", "c"=>"mela");
arsort ($frutta);
reset ($frutta);
while (list ($chiave, $valore) = each ($frutta)) {
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà

```
c = mela
d = limone
b = banana
a = arancia
```

I frutti sono ordinati in ordine alfabetico decrescente, e l'indice associato a ogni elemento è stato mantenuto.

È possibile modificare il comportamento dell'ordinamento usando il parametro opzionale `sort_flags`, per maggiori dettagli vedere sort().

vedere anche asort(), rsort(), ksort() e sort().

asort (PHP 3, PHP 4 >= 4.0.0)

Ordina un array e mantiene le associazioni degli indici

void **asort** (array array [, int sort_flags]) \linebreak

Questa funzione ordina un array in modo tale che i suoi indici mantengano la loro correlazione con gli elementi ai quali sono associati. Viene usata principalmente nell'ordinamento degli array associativi, quando la disposizione originaria degli elementi è importante .

Esempio 1. esempio di asort()

```
$frutta = array ("d"=>"limone", "a"=>"arancia", "b"=>"banana", "c"=>"mela");
asort ($frutta);
reset ($frutta);
while (list ($chiave, $valore) = each ($frutta)) {
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà:

```
a = arancia
b = banana
d = limone
c = mela
```

I frutti sono ordinati in ordine alfabetico, e l'indice associato ad ogni elemento è stato mantenuto.

È possibile modificare il comportamento dell'ordinamento usando il parametro opzionale `sort_flags`, per maggiori dettagli vedere sort().

Vedere anche arsort(), rsort(), ksort() e sort().

compact (PHP 4 >= 4.0.0)

Crea un array contenente variabili e il loro valore

array **compact** (mixed varname [, mixed ...]) \linebreak

compact() accetta un numero variabile di parametri. Ogni parametro può essere una stringa contenente il nome della variabile, o un array di nomi di variabile. L’array può contenere altri array di nomi di variabile; **compact()** se ne occupa in modo ricorsivo.

Per ognuno di questi, **compact()** cerca la variabile con quel nome nella tabella dei simboli corrente, e la aggiunge all’array di output in modo tale che il nome della variabile diventi la chiave e i contenuti della variabile diventino il valore associato a quella chiave. In breve, **compact()** è l’opposto di extract(). Restituisce l’array di output con tutte le variabili aggiunte a quest’ultimo.

Qualsiasi stringa non valorizzata verrà semplicemente ignorata.

Esempio 1. esempio di compact()

```
$citta = "Milano";
$provincia = "MI";
$evento = "SMAU";

$var_luoghi = array ("citta", "provincia");

$risultato = compact ("evento", "niente", $var_luoghi);
```

In questo modo, `$risultato` sarà:

```
Array
(
    [event] => SMAU
    [citta] => Milano
    [provincia] => MI
)
```

Vedere anche extract().

count (PHP 3, PHP 4 >= 4.0.0)

Conta gli elementi in una variabile

int **count** (mixed var) \linebreak

Restituisce il numero di elementi in *`var`*, la quale è di norma un array (dal momento che qualsiasi altro oggetto avrà un elemento).

Se `var` non è un array, verrà restituito `1` (eccezione: `count(NULL)` restituisce `0`).

> **Attenzione**
>
> **count()** può restituire 0 per una variabile che non è impostata, ma può anche restituire 0 per una variabile che è stata inizializzata con un array vuoto. Usare isset() per verificare se una variabile è impostata.

Vedere la sezione Arrays nel manuale per una spiegazione dettagliata di come gli array siano implementati ed usati in PHP.

Esempio 1. esempio di count()

```
$a[0] = 1;
$a[1] = 3;
$a[2] = 5;
$risultato = count ($a);
//$risultato == 3

$b[0] = 7;
$b[5] = 9;
$b[10] = 11;
$risultato = count ($b);
// $risultato == 3;
```

> **Nota:** La funzione sizeof() è un alias per **count()**.

Vedere anche is_array(), isset() e strlen().

current (PHP 3, PHP 4 >= 4.0.0)

Restituisce l'elemento corrente di un array

mixed **current** (array array) \linebreak

Ogni array ha un puntatore interno all'elemento "corrente", che è inizializzato al primo elemento inserito nell'array.

La funzione **current()** restituisce l'elemento che è attualmente puntato dal puntatore interno. In ogni caso non muove il puntatore. Se il puntatore interno punta oltre la fine della lista di elementi, **current()** restituisce `FALSE`.

> **Attenzione**
>
> Se l’array contiene elementi vuoti (0 o "", la stringa vuota) la funzione restituirà FALSE pure per questi elementi. Questo rende impossibile stabilire se si è veramente alla fine della lista in un array di questo tipo usando **current()**. Per attraversare in modo corretto un array che può contenere elementi vuoti, usare la funzione each().

Vedere anche end(), next(), prev() e reset().

each (PHP 3, PHP 4 >= 4.0.0)

Restituisce la successiva coppia chiave/valore di un array e incrementa il puntatore dell’array

array **each** (array array) \linebreak

Restituisce la corrente coppia chiave/valore corrente di `array` e incrementa il puntatore interno dell’array. Questa coppia è restituita in un array di quattro elementi, con le chiavi *0*, *1*, *key*, and *value*. Gli elementi *0* e *key* contengono il nome della chiave dell’elemento dell’array, mentre *1* e *value* contengono i dati.

Se il puntatore interno dell’array punta oltre la fine dei contenuti dell’array, **each()** restituisce `FALSE`.

Esempio 1. esempi dieach()

```
$foo = array ("bob", "fred", "jussi", "jouni", "egon", "marliese");
$bar = each ($foo);
```

`$bar` ora contiene la seguente coppia chiave/valore:
- 0 => 0
- 1 => ’bob’
- key => 0
- value => ’bob’

```
$foo = array ("Robert" => "Bob", "Seppo" => "Sepi");
$bar = each ($foo);
```

`$bar` ora contiene la seguente coppia chiave/valore:

- 0 => ’Robert’
- 1 => ’Bob’
- key => ’Robert’
- value => ’Bob’

each() viene normalmente usata in congiunzione con list() nell’attraversamento di un array; per esempio, `$HTTP_POST_VARS`:

Esempio 2. Attraversamento di `$HTTP_POST_VARS` con each()

```
echo "Valori inviati con il metodo POST:<br>";
reset ($HTTP_POST_VARS);
while (list ($chiave, $valore) = each ($HTTP_POST_VARS)) {
    echo "$chiave => $valore<br>";
}
```

Dopo l’esecuzione di **each()**, il puntatore dell’array viene lasciato sull’elemento successivo, o sull’ultimo elemento se si è alla fine dell’array. Si deve utilizzare reset() se si vuole riattraversare l’array usando **each()**.

Vedere anche key(), list(), current(), reset(), next(), prev() e **foreach()**.

end (PHP 3, PHP 4 >= 4.0.0)

Sposta il puntatore interno dell’array all’ultimo elemento

mixed **end** (array array) \linebreak

end() fa avanzare il puntatore di *`array`* all’ultimo elemento, e restituisce l’elemento.

Esempio 1. Un semplce esempio di end()

```
<?php

  $frutti = array('mela','banana','mirtillo');

  print end($frutti); // mirtillo

?>
```

Vedere anche current(), each(), next() e reset().

extract (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Importa le variabili nella tabella dei simboli

int **extract** (array var_array [, int extract_type [, string prefix]]) \linebreak

Questa funzione viene usata per importare delle variabili da un array nella tabella dei simbloi corrente. Riceve un array associativo *var_array* e interpreta le chiavi come nomi di variabile e i valori come valori di variabile. Per ogni coppia chiave/valore verrà creata una variabile nella tabella dei simboli corrente, coerentemente con i parametri *extract_type* e *prefix*.

Nota: Since version 4.0.5 this function returns the number of variables extracted.

Nota: EXTR_IF_EXISTS e EXTR_PREFIX_IF_EXISTS sono stati introdotti nella versione 4.2.0.

extract() controlla ogni chiave per stabilire se costituisce un nome valido di variabile e se ci sono collisioni con variabili già esistenti nella tabella dei simboli. Il modo in cui vengono trattate le chiavi invalide/numeriche e le collisioni è determinato da *extract_type*. Può essere uno dei seguenti valori:

EXTR_OVERWRITE

Se avviene una collisione, sovrascrive la variabile esistente.

EXTR_SKIP

Se avviene una collisione, non sovrascrive la variabile esistente.

EXTR_PREFIX_SAME

Se avviene una collisione, mette come prefisso al nome della variabile il parametro *prefix*.

EXTR_PREFIX_ALL

Mette come prefisso di tutte le variabili il parametro *prefix*. Dal PHP 4.0.5 questo avviene anche per i valori numerici.

EXTR_PREFIX_INVALID

Mette come prefisso, solo per i nomi di variabili invalidi/numerici, il parametro *prefix*. Questa opzione è stata aggiunta in PHP 4.0.5.

EXTR_IF_EXISTS

Sovrascrive la variabile solo se già esiste nella tabella dei simboli, altrimenti non fa nulla. Questo è utile per definire una lista di variabili valide e quindi estrarre solo quelle variabili definite in $_REQUEST, per esempio. Questa opzione è stata aggiunta in PHP 4.2.0.

EXTR_PREFIX_IF_EXISTS

Crea nomi di variabili con il prefisso solo se la versione senza prefisso della stessa variable esiste nella tabella dei simboli. Questa opzione è stata aggiunta in PHP 4.2.0.

Se *extract_type* non è specificato, si assume che sia EXTR_OVERWRITE.

Si noti che *prefix* è richiesto solo se *extract_type* è EXTR_PREFIX_SAME, EXTR_PREFIX_ALL, EXTR_PREFIX_INVALID o EXTR_PREFIX_IF_EXISTS. Se il risultato non è un nome di variabile valido, non viene importato nella tabella dei simboli.

extract() restituisce il numero di variabili importate con successo nella tabella dei simboli.

Un possibile uso di **extract()** è quello di importare nella tabella dei simboli variabili contenute in un array associativo restituito da wddx_deserialize().

Esempio 1. esempio diextract()

```
<?php

/* Si supponga che $array_variabili sia un array restituito da
   wddx_deserialize */

$dimensione = "grande";
$array_variabili = array ("colore" => "blu",
                          "dimensione"  => "media",
                          "forma" => "sfera");
extract ($array_variabili, EXTR_PREFIX_SAME, "wddx");

print "$colore, $dimensione, $forma, $wddx_dimensione\n";

?>
```

Questo esempio mostrerà:

```
blu, grande, sfera, media
```

La variabile `$dimensione` non è stata sovrascritta, in quanto è specificato EXTR_PREFIX_SAME, che ha portato alla creazione di `$wddx_dimensione`. Se fosse stato specificato EXTR_SKIP, $wddx_dimensione non sarebbe stata creata. EXTR_OVERWRITE avrebbe portato `$dimensione` ad assumere il valore "medio" , e EXTR_PREFIX_ALL avrebbe fatto creare nuove variabili chiamate `$wddx_colore`, `$wddx_dimensione` e `$wddx_forma`.

Si deve usare un array associativo, un array indicizzato numericamente non produce risultati a meno di non usare EXTR_PREFIX_ALL o EXTR_PREFIX_INVALID.

Vedere anche compact().

in_array (PHP 4 >= 4.0.0)

Restituisce `TRUE` se un valore è presente in un array

bool **in_array** (mixed ago, array pagliaio [, bool strict]) \linebreak

Cerca in *`pagliaio`* per trovare *`ago`* e restituisce `TRUE` se viene trovato nell'array, `FALSE` altrimenti.

Se il terzo parametro *strict* è TRUE la funzione **in_array()** controllerà anche il tipo di *ago* nell'array *haystack*.

> **Nota:** Se *ago* è una stringa, il confronto è effettuato tenendo conto delle maiuscole/minuscole.

> **Nota:** Nelle versioni di PHP precedenti la 4.2.0. *ago* non poteva essere un array.

Esempio 1. esempio di in_array()

```
$os = array ("Mac", "NT", "Irix", "Linux");
if (in_array ("Irix", $os)) {
    print "Trovato Irix";
}
if (in_array ("mac", $os)) {
    print "Trovato mac";
}
```

La seconda condizione fallisce perché **in_array()** tiene conto di maiuscole e minuscole, quindi il programma mostrerà:

```
Trovato Irix
```

Esempio 2. esempio di in_array() con strict

```
<?php
$a = array('1.10', 12.4, 1.13);

if (in_array('12.4', $a, TRUE))
    echo "'12.4' trovato con controllo strict\n"
if (in_array(1.13, $a, TRUE))
    echo "1.13 trovato con controllo strict\n"
?>
```

Questo mostrerà:

```
1.13 trovato con controllo strict
```

Esempio 3. in_array() con un array come ago

```
<?php
$a = array(array('p', 'h'), array('p', 'r'), 'o');

if (in_array(array ('p', 'h'), $a))
    echo "'ph' trovato\n";
if (in_array(array ('f', 'i'), $a))
    echo "'fi' non trovato\n";
if (in_array('o', $a))
    echo "'o' trovato\n";
?>

// Questo ritornerà:

'ph' trovato
'o' trovato
```

Vedere anche array_search().

key (PHP 3, PHP 4 >= 4.0.0)

Estrae la chiave corrente da un array associativo

mixed **key** (array array) \linebreak

key() restituisce la chiave corrispondente all'attuale posizione del puntatore interno all'array.

Vedere anche current() e next().

krsort (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Ordina rispetto alle chiavi di un array in ordine inverso

int **krsort** (array array [, int sort_flags]) \linebreak

Ordina un array rispetto alle sue chiavi, in ordine inverso, mantenendo le associazioni. Questa funzione è utile con gli array associativi.

Esempio 1. Esempio di krsort()

```
$frutti = array ("d"=>"limone", "a"=>"arancio", "b"=>"banana", "c"=>"mela");
krsort ($frutti);
reset ($frutti);
while (list ($chiave, $valore) = each ($frutti)) {
```

```
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà:

```
d = limone
c = mela
b = banana
a = arancio
```

Si può modificare il comportamento dell'ordinamento usando il parametro opzionale `sort_flags`, per ulteriori dettagli vedere sort().

Vedere anche asort(), arsort(), ksort(), sort(), natsort() e rsort().

ksort (PHP 3, PHP 4 >= 4.0.0)

Ordina rispetto alle chiavi di un array

int **ksort** (array array [, int sort_flags]) \linebreak

Ordina un array rispetto alle sue chiavi, mantenendo le associazioni. Questa funzione è utile con gli array associativi.

Esempio 1. esempio di ksort()

```
$frutti = array ("d"=>"limone", "a"=>"arancia", "b"=>"banana", "c"=>"mela");
ksort ($frutti);
reset ($frutti);
while (list ($chiave, $valore) = each ($frutti)) {
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà:

```
a = arancia
b = banana
c = mela
d = limone
```

Si può modificare il comportamento dell'ordinamento usando il parametro opzionale `sort_flags`, per ulteriori dettagli vedere sort().

Vedere anche asort(), arsort(), krsort(), uksort(), sort(), natsort() e rsort().

> **Nota:** Il secondo parametro è stato aggiunto in PHP 4.

list (unknown)

Assegna valori a delle variabili come se fossero un array

void **list** (mixed ...) \linebreak

Come array(), questa non è in realtà una funzione, bensì un costrutto del linguaggio. **list()** è usata per assegnare valori ad una lista di variabili in una sola operazione.

> **Nota: list()** funziona solo su array numerici e si aspetta che gli indici numerici partano da 0.

Esempio 1. esempio di list()

```
<?php

$info = array('caffè', 'scuro', 'caffeina');

// assegna a tutte le variabili
list($bevanda, $colore, $componente) = $info;
print "Il $bevanda è $colore e la $componente lo rende speciale.\n";

// assegna solo in parte
list($bevanda, , $componente) = $info;
print "Il $bevanda ha la $componente.\n";

// oppure assegnamo solo l'ultima variabile
list( , , $componente) = $info;
print "Ho voglia di $bevanda!\n";

?>
```

Esempio 2. Esempio di uso di list()

```
<table>
 <tr>
  <th>Nome dell'impiegato</th>
  <th>Stipendio</th>
 </tr>

<?php

$risultato = mysql_query ($conn, "SELECT id, nome, stipendio FROM impiegati",$conn);
while (list ($id, $nome, $stipendio) = mysql_fetch_row ($risultato)) {
    print (" <tr>\n".
           "  <td><a href=\"info.php?id=$id\">$nome</a></td>\n".
           "  <td>$stipendio</td>\n".
           " </tr>\n");
}

?>

</table>
```

Vedere anche each() e array() e extract().

natcasesort (PHP 4 >= 4.0.0)

Ordina un array usando un algoritmo di "ordine naturale" non sensibile alle maiuscole/minuscole

void **natcasesort** (array array) \linebreak

Questa funziona implementa un algoritmo di ordinamento che ordina le stringhe alfanumeriche come lo farebbe un essere umano. Questo è chiamato "ordine naturale".

natcasesort() è una versione, non sensibile alle maiuscole/minuscole, di natsort(). Vedere natsort() per un esempio della differenza tra questo algoritmo e quello normalmente usato dai computer.

Per maggiori informazioni vedere la pagina di Martin Pool Natural Order String Comparison (http://naturalordersort.org/) .

Vedere anche sort(), natsort(), strnatcmp() e strnatcasecmp().

natsort (PHP 4 >= 4.0.0)

Ordina un array usando un algoritmo di "ordine naturale"

void **natsort** (array array) \linebreak

Questa funzione implementa un algoritmo di ordinamento che ordina le stringhe alfanumeriche come lo farebbe un essere umano. Questo è chiamato "ordine naturale". un esempio della differenza tra questo algoritmo e quello normalmente usato dai computer (usato in sort()) è dato qui sotto:

Esempio 1. esempio di natsort()

```
$array1 = $array2 = array ("img12.png", "img10.png", "img2.png", "img1.png");

sort($array1);
echo "Ordinamento standard\n";
print_r($array1);

natsort($array2);
echo "\nOrdinamento naturale\n";
print_r($array2);
```

Questo codice genererà il seguente risultato:

```
Ordinamento standard
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Ordinamento naturale
Array
(
    [3] => img1.png
    [2] => img2.png
    [1] => img10.png
    [0] => img12.png
)
```

Per ulteriori informazioni vedere la pagina di Martin Pool Natural Order String Comparison (http://naturalordersort.org/) .

Vedere anche natcasesort(), strnatcmp() e strnatcasecmp().

next (PHP 3, PHP 4 >= 4.0.0)

Incrementa il puntatore interno dell’array

mixed **next** (array array) \linebreak

Restituisce l'elemento dell'array che sta nella posizione successiva a quella attuale indicata dal puntatore interno, oppure FALSE se non ci sono altri elementi.

next() si comporta come current(), con una differenza. Incrementa il puntatore interno dell'array di una posizione, prima di restituire l'elemento. Ciò significa che restituisce l'elemento successivo e incrementa il puntatore di una posizione. Se l'incremento fa sì che il puntatore vada oltre la fine della lista di elementi, **next()** restituisce FALSE.

> **Attenzione**
>
> Se l'array contiene elementi vuoti, o elementi che hanno il valore chiave uguale a 0 allora questa funzione restituisce FALSE anche per questi elementi. Per esplorare correttamente un array che può contenere elementi vuoti o con chiave uguale a 0 vedere la funzione each().

Vedere anche current(), end(), prev() e reset().

pos (PHP 3, PHP 4 >= 4.0.0)

Restituisce l'elemento corrente di un array

mixed **pos** (array array) \linebreak

Questo è un alias di current().

Vedere anche end(), next(), prev() e reset().

prev (PHP 3, PHP 4 >= 4.0.0)

Decrementa il puntatore interno dell'array

mixed **prev** (array array) \linebreak

Restituisce l'elemento dell'array che sta nella posizione precedente a quella attuale indicata dal puntatore interno, oppure FALSE se non ci sono altri elementi.

> **Attenzione**
>
> Se l'array contiene degli elementi vuoti la funzione restituirà FALSE per questi valori. Per esplorare correttamente un array che può contenere elementi vuoti vedere la funzione each().

prev() si comporta come next(), tranne per il fatto di decrementare il puntatore interno di una posizione, invece che incrementarlo.

Vedere anche current(), end(), next() e reset().

range (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Crea un array contenente una serie di elementi

array **range** (mixed min, mixed max) \linebreak

range() restituisce una serie di elementi da `min` a `max`, inclusiva. Se `min` > `max`, la sequenza sarà decrescente.

Esempio 1. esempi di range()

```
foreach(range(0, 9) as $numero) {
    echo $numero;
}
foreach(range('a', 'z') as $lettera) {
    echo $lettera;
}
foreach(range('z', 'a') as $lettera) {
    echo $lettera;
}
```

> **Nota:** Prima della versione 4.1.0 la funzione **range()** generava solo array crescenti di interi. Il supporto per le sequenze di caratteri e array decresenti è stata aggiunta nella 4.1.0.
>
> **Esempio 2. Simulazione di range decrescenti e sequenze di caratteri**
>
> ```
> # array_reverse può essere udata per invertire l'ordine di un range
> foreach(range(0,9) as $numero) {
>     echo $numero;
> }
>
> #array_map() puà essere udata per convertire gli interi in caratteri usando chr()
> foreach(array_map('chr', range(ord('a'),ord('z'))) as $carattere) {
>     echo $carattere;
> }
> ```

Vedere shuffle() per un altro esempio d'uso.

reset (PHP 3, PHP 4 >= 4.0.0)

Reimposta il puntatore interno di un array sulla posizione iniziale

mixed **reset** (array array) \linebreak

reset() riporta il puntatore di `array` sul primo elemento.

reset() restituisce il volore del primo elemento dell'array.

Vedere anche current(), each(), next(), e prev().

rsort (PHP 3, PHP 4 >= 4.0.0)

Ordina un array in ordine decrescente

void **rsort** (array array [, int sort_flags]) \linebreak

Questa funzione ordina un array in ordine decrescente.

Esempio 1. esempio di rsort()

```
$frutti = array ("limone", "arancia", "banana", "mela");
rsort ($frutti);
reset ($frutti);
while (list ($chiave, $valore) = each ($frutti)) {
    echo "$chiave = $valore\n";
}
```

Questo esempio mostrerà:

```
0 = mela
1 = limone
2 = banana
3 = arancia
```

I frutti sono stati ordinati in ordine alfabetico decrescente.

Si può modificare il comportamento dell'ordinamento usando il parametro opzionale *sort_flags*, per maggiori dettagli vedere sort().

Vedere anche arsort(), asort(), ksort(), sort() e usort().

shuffle (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Mescola un array

void **shuffle** (array array) \linebreak

Questa funzione mescola un array (rende casuale l'ordine degli elementi). Si deve usare srand() per inizializzare il generatore di numeri casuali.

Esempio 1. esempio di shuffle()

```
$numeri = range (1,20);
srand ((float)microtime()*1000000);
shuffle ($numeri);
while (list (, $numeero) = each ($numeri)) {
    echo "$numero ";
}
```

Vedere anche arsort(), asort(), ksort(), rsort(), sort() e usort().

sizeof (PHP 3, PHP 4 >= 4.0.0)

Conta gli elementi in una variabile

int **sizeof** (mixed var) \linebreak

La funzione **sizeof()** è un alias di count().

Vedere anche count().

sort (PHP 3, PHP 4 >= 4.0.0)

Ordina un array

void **sort** (array array [, int sort_flags]) \linebreak

Questa funzione ordina un array. Gli elementi vengono disposti dal più piccolo al più grande.

Esempio 1. esempio di sort()

```
<?php

$frutti = array ("limone", "arancia", "banana", "mela");
sort ($frutti);
reset ($frutti);
while (list ($chiave, $valore) = each ($frutti)) {
    echo "frutti[".$chiave."] = ".$valore."\n";
}

?>
```

Questo esempio mostrerà:

```
frutti[0] = arancia
frutti[1] = banana
frutti[2] = limone
frutti[3] = mela
```

I frutti sono stati ordinati in ordine alfabetico.

Il secondo parametro opzionale *sort_flags* può essere usato per modificare il comportamento dell’ordinamento, usando i seguenti valori:

flag d’ordinamento:

- SORT_REGULAR - compara gli elementi in modo normale
- SORT_NUMERIC - compara gli elementi numericamente
- SORT_STRING - compara gli elementi convertiti in stringa

Vedere anche arsort(), asort(), ksort(), natsort(), natcasesort(), rsort(), usort(), array_multisort() e uksort().

> **Nota:** Il secondo parametro è stato aggiunto nel PHP 4.

uasort (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Ordina un array mediante una funzione definita dall’utente e mantiene le associazioni

void **uasort** (array array, function cmp_function) \linebreak

Questa funzione ordina un array in modo tale che le chiavi mantengano la loro correlazione con gli elementi dell’array a cui sono associate. Questo è utile quando si ordinano array associativi in cui l’ordine degli elementi è importante. La funzione di comparazione deve essere fornita dall’utente.

> **Nota:** Vedere usort() e uksort() per esempio di funzioni di comparazione.

> **Nota:** Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

Vedere anche usort(), uksort(), sort(), asort(), arsort(), ksort() e rsort().

uksort (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Ordina rispetto alle chiavi di un array mediante una funzione definita dall'utente

void **uksort** (array array, function cmp_function) \linebreak

Ordina rispetto alle chiavi di un array mediante una funzione di comparazione definita dall'utente. Se si vuole ordinare un array con dei criteri non usuali, si deve usare questa funzione.

Esempio 1. esempio di uksort()

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (4 => "quattro", 3 => "tre", 20 => "venti", 10 => "dieci");

uksort ($a, "cmp");

while (list ($chiave, $valore) = each ($a)) {
    echo "$chiave: $valore\n";
}
```

Questo esempio mostrerà:

```
20: venti
10: dieci
4: quattro
3: tre
```

Nota: Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

Vedere anche usort(), uasort(), sort(), asort(), arsort(), ksort(), natsort() e rsort().

usort (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Ordina un array mediante una funzione definita dall'utente

void **usort** (array array, string cmp_function) \linebreak

Ordina i valori di un array mediante una funzione di comparazione definita dall'utente. Se si vuole ordinare un array con dei criteri non usuali, si deve usare questa funzione.

La funzione di comparazione deve restituire un intero minore, uguale o superiore a zero se il primo elemento è da considerarsi rispettivamente monire, uguale o maggiore del secondo. Se due parametri vengono valutati uguale, il loro ordinamento nell'array ordinato è indefinito..

Esempio 1. esempio di usort()

```
function cmp ($a, $b) {
    if ($a == $b) return 0;
    return ($a > $b) ? -1 : 1;
}

$a = array (3, 2, 5, 6, 1);

usort ($a, "cmp");

while (list ($chiave, $valore) = each ($a)) {
    echo "$chiave: $valore\n";
}
```

Questo esempio mostrerà:

```
0: 6
1: 5
2: 3
3: 2
4: 1
```

> **Nota:** Ovviamente, in questo caso banale di ordinamento decrescente la funzione rsort() sarebbe stata più appropriata.

Esempio 2. esempio di usort() con un array multidimensionale

```
function cmp ($a, $b) {
    return strcmp($a["frutto"], $b["frutto"]);
}

$frutti[0]["frutto"] = "limoni";
$frutti[1]["frutto"] = "arance";
$frutti[2]["frutto"] = "uva";

usort($frutti, "cmp");
```

```
while (list ($chiave, $valore) = each ($frutti)) {
    echo "\$frutti[$chiave]: " . $valore["frutto"] . "\n";
}
```

Quando si ordina un array multidimensionale, $a e $b contengono riferimenti al primo indice dell'array.

Questo esempio mostrerà:

```
$frutti[0]: arance
$frutti[1]: limoni
$frutti[2]: uva
```

> **Nota:** Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

Esempio 3. esempio di usort() usando una funzione membro di un oggetto

```
class OggettoTest {
    var $nome;

    function OggettoTest($nome)
    {
        $this->nome = $nome;
    }

    /* Questa &egrave; la funzione statica di comparazione: */
    function comp_ogg($a, $b)
    {
        $al = strtolower($a->nome);
        $bl = strtolower($b->nome);
        if ($al == $bl) return 0;
        return ($al > $bl) ? +1 : -1;
    }
}

$a[] = new OggettoTest("c");
$a[] = new OggettoTest("b");
$a[] = new OggettoTest("d");

uasort($a, array ("OggettoTest", "comp_ogg"));

foreach ($a as $voce) {
    print $voce->nome."\n";
}
```

Questo esempio mostrerà:

```
b
c
d
```

Attenzione

La sottostante funzione di quicksort può causare, in alcune librerie C (per esempio, sui sistemi Solaris) un crash del PHP se la funzione di comparazione non restituisce valori coerenti.

Vedere anche uasort(), uksort(), sort(), asort(), arsort(),ksort(), natsort() e rsort().

III. Funzioni Aspell [deprecated]

Introduzione

Le funzioni **aspell()** permettono di controllare la correttezza di una parola e di offrire suggerimenti.

Requisiti

aspell funziona solo con versioni molto vecchie (più o meno fino alla .27.*) della libreria aspell. Né il presente modulo, né quelle versioni della libreria sono più supportate. Se si vuole usare le funzionalità di controllo grammaticale in php, usare piuttosto pspell. Usa la libreria pspell e funziona con le nuove versioni di aspell.

Istallazione

È necessaria la libreria aspell, disponibile su: http://aspell.sourceforge.net/.

Vedere Anche

Vedere anche pspell.

aspell_check (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Controlla una parola [deprecated]

bool **aspell_check** (int link_dizionario, string parola) \linebreak

aspell_check() controlla la compitazione di una parola e restituisce `TRUE` se è corretta, `FALSE` altrimenti.

Esempio 1. aspell_check()

```
$aspell_link = aspell_new("italiano");

if (aspell_check($aspell_link, "provva")) {
    echo "La parola &egrave; corretta";
} else {
    echo "Spiacente, parola non corretta";
}
```

aspell_check_raw (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Controlla una parola senza togliere le maiuscole o cercare di eliminare gli spazi inutili [deprecated]

bool **aspell_check_raw** (int link_dizionario, string parola) \linebreak

aspell_check_raw() controlla la correttezza di una parola, senza modificare le maiuscole/minusciole o cercare di eliminare gli spazi inutili e restituisce `TRUE` se è corretta, `FALSE` altrimenti.

Esempio 1. aspell_check_raw()

```
$aspell_link = aspell_new("italiano");

if (aspell_check_raw($aspell_link, "prova")) {
    echo "La parola &grave; corretta";
} else {
    echo "Spiacente, parola non corretta";
}
```

aspell_new (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Carca un nuovo dizionario [deprecated]

int **aspell_new** (string master [, string personal]) \linebreak

aspell_new() apre un nuovo dizionario e restituisce un puntatore (link) identificatore del dizionario, da utilizzare in altre funzioni aspell. Restituisce `FALSE` in caso di errore.

Esempio 1. aspell_new()

```
$aspell_link = aspell_new("italiano");
```

aspell_suggest (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Suggerisce correzioni di una parola [deprecated]

array **aspell_suggest** (int link_dizionario, string parola) \linebreak

aspell_suggest() restituisce un array di possibili correzioni per la parola data.

Esempio 1. aspell_suggest()

```
$aspell_link = aspell_new("italiano");

if (!aspell_check($aspell_link, "prova")) {
    $suggerimenti = aspell_suggest($aspell_link, "prova");

    foreach ($suggerimenti as $suggerimento) {
        echo "Possibile parola corretta: $suggerimento<br>\n";
    }
}
```

IV. Funzioni Matematiche BCMath a precisione arbitraria

Introduzione

Per la matematica a precisione arbitraria PHP offre il Binary Calculator che supporta numeri di qualsiasi dimensione e precisione, rappresentati da stringhe;

Requisiti

Dalla versione 4.0.4 del PHP, libbcmath è inclusa nella distribuzione. Non c'è bisogno di altre librerie esterne per questa estensione.

Istallazione

In PHP 4, queste funzioni sono disponibili solo se il PHP è stato configurato con `--enable-bcmath`. In PHP 3, queste funzioni sono disponibili solo se il PHP non è stato configurato con `--disable-bcmath`.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

Questa estensione non definisce alcun tipo di risorsa.

Costanti Predefinite

Questa estensione non definisce alcuna costante.

bcadd (PHP 3, PHP 4 >= 4.0.0)

Somma due numeri a precisione arbitraria

string **bcadd** (string primo operando, string secondo operando [, int precisione]) \linebreak

Somma il *`primo operando`* con il *`secondo operando`* e restituisce la somma sotto forma di stringa. Il parametro opzionale *`precisione`* è utilizzato per impostare il numero di cifre dopo il punto decimale nel risultato.

Vedere anche bcsub().

bccomp (PHP 3, PHP 4 >= 4.0.0)

Confronta due numeri a precisione arbitraria

int **bccomp** (string primo operand, string secondo operand [, int precisione]) \linebreak

Confronta il *`primo operando`* e il *`secondo operando`* e restituisce il risultato sotto forma di intero. Il parametro opzionale *`precisione`* è utilizzato per impostare il numero di cifre dopo il punto decimale che verranno usate nel confronto. Il valore restituito ‘ 0 se i due operandi sono uguali. Se il *`primo operando`* è più grande del *`secondo operando`* il valore restituito è +1 e se il *`primo operando`* è minore del *`secondo operando`* il valore restituito è -1.

bcdiv (PHP 3, PHP 4 >= 4.0.0)

Divide due numeri a precisione arbitraria

string **bcdiv** (string primo operando, string secondo operando [, int precisione]) \linebreak

Divide il *`primo operando`* per il *`secondo operando`* e restituisce il risultato. Il parametro opzionale *`precisione`* imposta il numero di cifre dopo il punto decimale nel risultato.

Vedere anche bcmul().

bcmod (PHP 3, PHP 4 >= 4.0.0)

Ricava il modulo di un numero a precisione arbitraria

string **bcmod** (string operando, string modulo) \linebreak

Ricava il modulo di *`operando`* usando *`modulo`*.

Vedere anche bcdiv().

bcmul (PHP 3, PHP 4 >= 4.0.0)

Moltiplica due numeri a precisione arbitraria

string **bcmul** (string primo operando, string secondo operando [, int precisione]) \linebreak

Moltiplica il *primo operando* per il *secondo operando* e restituisce il risultato. Il parametro opzionale *precisione* imposta il numero di cifre dopo il punto decimale nel risultato.

Vedere anche bcdiv().

bcpow (PHP 3, PHP 4 >= 4.0.0)

Effettual l'elevamento a potenza

string **bcpow** (string x, string y [, int precisione]) \linebreak

Eleva *x* alla potenza *y*. Il parametro opzionale *precisione* può essere usato per imposta il numero di cifre dopo il punto decimale nel risultato.

Vedere anche bcsqrt().

bcscale (PHP 3, PHP 4 >= 4.0.0)

Imposta il valore di precisione di default per tutte le funzioni matematich BCMath

string **bcscale** (int precisione) \linebreak

Questa funzione imposta il valore di default del parametro *precisione* per tutte le funzioni BCMath susseguenti, che non specifichino esplicitamente un parametro di precisione numerica.

bcsqrt (PHP 3, PHP 4 >= 4.0.0)

Ottiene la radice quadrata di un numero a precisione arbitraria

string **bcsqrt** (string operando [, int precisione]) \linebreak

Restituisce la radice quadrata di *operando*. Il parametro opzionale *precisione* imposta il numero di cifre dopo il punto decimale nel risultato.

Vedere anche bcpow().

bcsub (PHP 3, PHP 4 >= 4.0.0)

Sottrae un numero a precisione arbitraria da un altro

string **bcsub** (string primo operando, string secondo operando [, int precisione]) \linebreak

Sottrae il `primo operando` dal `secondo operando` e retituisce il risultato in una stringa. Il parametro opzionale `scale` è usato per impostare il numero di cifre dopo il punto decimale nel risultato.

Vedere anche bcadd().

V. Funzioni di compressione Bzip2

Introduzione

Le funzioni bzip2 sono utilizzate per leggere e scrivere in modo trasparente i file compressi con bzip2 (.bz2).

Requisiti

Questo modulo tuilizza le funzioni della libreria bzip2 (http://sources.redhat.com/bzip2/) di Julian Seward

Istallazione

Il supporto di Bzip2 in PHP non è attivo per default. Si deve usare l'opzione di configurazione --with-bz2 quando si compila PHP per sbilitare il supporto bzip2. Questo modulo richiede bzip2/libbzip2 con versione >= 1.0.x.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

Questa estensione definisce un tipi di risorsa: un puntatore a file che identifica il file bz2 su cui lavorare.

Costanti Predefinite

Questa estensione non definisce alcuna costante.

Esempi

Questo esempio apre un file temporaneo e scrive una stringa di prova su di esso, quindi stampa il contenuto del file.

Esempio 1. breve esempio di bzip2

```
<?php
```

```
$nomefile = "/tmp/filediprova.bz2";
$str = "Questa è una stringa di prova.\n";

// apre il file in lettura
$bz = bzopen($nomefile, "w");

// scrive la stringa sul file
bzwrite($bz, $str);

// chiude il file
bzclose($bz);

// apre il file in lettura
$bz = bzopen($nomefile, "r");

// legge 10 caratteri
print bzread($bz, 10);

// stampa fino alla fine del file (o fino ai prossimi 1024 caratteri) e chi-
ude il file.
print bzread($bz);

bzclose($bz);

?>
```

bzclose (PHP 4 >= 4.0.4)

Chiude un puntatore a un file bzip2

int **bzclose** (resource bz) \linebreak

Chiude il file bzip2 referenziato dal puntatore *bz*.

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

Il puntatore al file deve essere valido, e deve puntare a un file gi$agrave; aperto con bzopen().

Vedere anche bzopen().

bzcompress (PHP 4 >= 4.0.4)

Comprime una stringa nel formato bzip2

string **bzcompress** (string sorgente [, int dimblocco [, int workfactor]]) \linebreak

bzcompress() comprime la stringa *sorgente* e la restituisce come dati codificati in bzip2.

Il parametro opzionale *dimblocco* specifica la dimensione del blocco usato durante la compressione e dovrebbe essere un numero tra 1 e 9 dove 9 dà la compressione migliore, ma usando più risorse. *dimblocco* ha come valore predefinito 4.

Il parametro opzionale *workfactor* controlla il comportamento della fase di compressione quando deve trattare col caso peggiore, ovvero dati in ingresso molto ripetitivi. Il valore può variare tra 0 e 250, dove 0 è un caso speciale e 30 egrave; il valore di default. Indipendentemente dal parametro *workfactor*, i dat generati sono gli stessi.

Esempio 1. Esempio di bzcompress()

```
<?php
$str = "dati di prova";
$bzstr = bzcompress($str, 9);
print( $bzstr );
?>
```

See also bzdecompress().

bzdecompress (PHP 4 >= 4.0.4)

Decomprime dati codificati con bzip2

string **bzdecompress** (string sorgente [, int small]) \linebreak

bzdecompress() decomprime la strina *sorgente* contenente dati codificati in bzip2 e li restituisce. Se il parametro opzionale *small* è TRUE, verrà usato un algoritmo di decompressione alternativo che richiede meno memoria (la maximum quantità massima di meoria richiesta scende a 2300K) ma funziona a circa la metà della velocità. Vedere la documentazione di bzip2 (http://sources.redhat.com/bzip2/) per maggiori informazioni su questa funzionalità.

Esempio 1. bzdecompress()

```
<?php
$stringa_iniziale = "Sto facendo il mio lavoro?";
$bzstr = bzcompress($start_str);

print( "Stirnga Compressa: " );
print( $bzstr );
print( "\n<br>n" );

$stringa = bzdecompress($bzstr);
print( "Stringa Decompressa: " );
print( $str );
print( "\n<br>n" );
?>
```

See also bzcompress().

bzerrno (PHP 4 >= 4.0.4)

Restituisce il codice d'errore bzip2

int **bzerrno** (resource bz) \linebreak

Restituisce il codice di un qualsiasi errore bzip2 restituito dal puntatore al file *bz*.

Vedere anche bzerror() e bzerrstr().

bzerror (PHP 4 >= 4.0.4)

Restituisce il codice d'errore bzip2 e la stringa corrispondente in un array

array **bzerror** (resource bz) \linebreak

Restituisce il codice e la stringa di errore, sotto forma di array associativo, di un errore bzip2 restituito dal puntatore *bz*.

Esempio 1. Esempio di bzerror()

```
<?php
$errore = bzerror($bz);

echo $errore["errno"];
echo $errore["errstr"];
?>
```

Vedere anche bzerrno() e bzerrstr().

bzerrstr (PHP 4 >= 4.0.4)

restituisce la stringa di errore bzip2

string **bzerrstr** (resource bz) \linebreak

Resituisce la stringa di errore bzip2 restituito dal puntatore *bz*.

Vedere anche bzerrno() e bzerror().

bzflush (PHP 4 >= 4.0.4)

Forza la scrittura di tutti i dati nel buffer

int **bzflush** (resource bz) \linebreak

Forza la scrittura di tutti i dati che sono nel buffer del puntatore *bz*.

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

Vedere anche bzread() e bzwrite().

bzopen (PHP 4 >= 4.0.4)

Apre un file compresso bzip2

resource **bzopen** (string nomefile, string modo) \linebreak

Apre un file bzip2 (.bz2) in lettura o scrittura. *nomefile* è il nome del file da aprire. Il parametro *modo* egrave; simile a quello della funzione fopen() (‘r’ per lettura, ‘w’ per scrittura, ecc.).

Se l’operazione fallisce, la funzione restituisce FALSE, altrimenti restituisce un puntatore al file appena aperto.

Esempio 1. Esempio dibzopen()

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$file_decompresso = bzread($bz, filesize("/tmp/foo.bz2"));
bzclose($bz);

print( "Il contenuto di /tmp/foo.bz2 è: " );
print( "\n<br>n" );
print( $file_decompresso );
?>
```

Vedere anche bzclose().

bzread (PHP 4 >= 4.0.4)

Esegue la lettura binaria di un file bzip2

string **bzread** (resource bz [, int lunghezza]) \linebreak

bzread() legge fino a `lunghezza` byte dal puntatore bzip2 specificato da `bz`. La pettura termina quando `lunghezza` byte (decompressi) sono stati letti o quando viene raggiunto l'EOF. Se il parametro opzionale `lunghezza` è omesso, **bzread()** leggerà 1024 byte (decompressi) ogni volta.

Esempio 1. Esempio di bzread()

```
<?php
$bz = bzopen("/tmp/foo.bz2", "r");
$str = bzread($bz, 2048);
print( $str );
?>
```

Vedere anche bzwrite() e bzopen().

bzwrite (PHP 4 >= 4.0.4)

Esegue la scrittura binaria di un file bzip2

int **bzwrite** (resource bz, string dati [, int lunghezza]) \linebreak

bzwrite() scrie il contenuto della stringa `dati` nel file bzip2 puntato da `bz`. Se il parametro opzionale `lunghezza` è specificato, la scrittura si fermerà dopo che siano stati scritti `lunghezza` byte (decompressi) o al raggiungimento della fine della stringa.

Esempio 1. Esempio di bzwrite()

```
<?php
$str = "dati non compressi";
$bz = bzopen("/tmp/foo.bz2", "w");
bzwrite($bz, $str, strlen($str));
bzclose($bz);
?>
```

Vedere anche bzread() e bzopen().

VI. Funzioni Calendar

Introduzione

L'estensione calendar presenta una serie di funzioni che semplificano la conversione tra differenti formati di calendario. Il formato intermedio o standard è basato sul Conteggio del Giorno Giuliano. Il Conteggio Giuliano è un conteggio di giorni che parte molto prima di qualsiasi data la maggior parte della gente potrebbe usare (circa il 4000 a.C.). Per convertire tra i sistemi di calendario, si deve prima convertire nel sistema del Giorno Giuliano, poi nel sistema di calendario scelto. Il Conteggio del Giorno Giuliano è molto diverso dal Calendario Giulano! Per maggiori informazioni sui sistemi di calendario vedere http://genealogy.org/~scottlee/cal-overview.html. Parti di questa pagina sono inclusi in queste istruzioni, citate tra virgolette.

Istallazione

Per rendere disponibili queste funzioni, occorre compilare PHP con `--enable-calendar`.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

Questa estensione non definisce alcun tipo di risorsa.

Costanti Predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata

compilata nel PHP o se è stata caricata dinamicamente a runtime.

`CAL_GREGORIAN` (integer)

`CAL_JULIAN` (integer)

`CAL_JEWISH` (integer)

`CAL_FRENCH` (integer)

`CAL_NUM_CALS` (integer)

`CAL_DOW_DAYNO` (integer)

`CAL_DOW_SHORT` (integer)

`CAL_DOW_LONG` (integer)

`CAL_MONTH_GREGORIAN_SHORT` (integer)

`CAL_MONTH_GREGORIAN_LONG` (integer)

`CAL_MONTH_JULIAN_SHORT` (integer)

`CAL_MONTH_JULIAN_LONG` (integer)

`CAL_MONTH_JEWISH` (integer)

`CAL_MONTH_FRENCH` (integer)

cal_days_in_month (PHP 4 >= 4.1.0)

Restituisce il numero di giorni di un mese per un dato anno e calendario

int **cal_days_in_month** (int calendario, int mese, int anno) \linebreak

Questa funzione restituisce il numero di giorni coe compongono il `mese` dell'`anno` nel `calendar` specificato.

Vedere anche jdtounix().

cal_from_jd (PHP 4 >= 4.1.0)

Converte dal Giorno Giuliano ad un calendario e restituisce informazioni estese

array **cal_from_jd** (int giornogiuliano, int calendario) \linebreak

cal_info (PHP 4 >= 4.1.0)

Restituisce informazioni su un particolare calendario

array **cal_info** (int calendario) \linebreak

cal_to_jd (PHP 4 >= 4.1.0)

Converte da un calendario a un Giorno Giuliano

int **cal_to_jd** (int calendario, int mese, int giorno, int anno) \linebreak

easter_date (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Restituisce un timestamp UNIX della mezzanotte del giorno di Pasqua di un dato anno

int **easter_date** (int anno) \linebreak

Restituisce il timestamp UNIX corrispondente alla mezzanotte del giorno di Pasqua dell'anno specificato. Se non è specificato l'anno, si assume l'anno corrente.

Attenzione: Questa funzione gerererà un allarme (warning) se l'anno è fuori dall'escursione di validità dei timestamp UNIX (cioè prima del 1970 o dopo il 2037).

Esempio 1. esempio di easter_date()

```
echo date ("M-d-Y", easter_date(1999));        /* "Apr-04-1999" */
echo date ("M-d-Y", easter_date(2000));        /* "Apr-23-2000" */
echo date ("M-d-Y", easter_date(2001));        /* "Apr-15-2001" */
```

La data della Pasqua fu definita dal Concilio di Nicea nel 325 d.C. come la Domenica successiva alla prima luna piena dopo l'Equinozio di Primavera. Si assume che l'Equinozio cada sempre il 21 Marzo, quindi il calcolo si riduce alla determinazione della data della luna piena e la data della Domenica seguente. L'algoritmo qui usato fu proposto attorno all'anno 532 d.C. da Dionysius Exiguus (Dionigi il Piccolo). Nel Calendario Giuliano (for years before 1753) un semplice ciclo di 19 anni è usato per traciare le fasi della Luna. Nel Calendario Gregoriano (per gli anni dopo il 1753 - ideato da Clavius e Lilius, e introdotto da Papa Gregorio XIII nell'Ottobre 1582, e in Gran Bretagna e nelle sue colonie nel Settembre 1752) due fattori correttivi sono aggiunti per rendere più accurato il ciclo.

(Il codice è basato su un programma in C di Simon Kershaw, <webmaster@ely.anglican.org>)

Vedere easter_days() per il calcolo della Pasqua prima del 1970 o dopo il 2037.

easter_days (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Restituisce il numero di giorni tra il 21 Marzo e Pasqua, dato un anno

int **easter_days** (int anno) \linebreak

Restituisce il numero di giorni tra il 21 Marzo e Pasqua per un dato anno. Se l'ano non è specificato, si assume l'anno corrente.

Questa funzione può essere usata al posto di easter_date() per calcolare la Pasqua per gli anni che cadono fuori dalla gamma di validità dei timestamp UNIX (cioè prima del 1970 o dopo il 2037).

Esempio 1. esempio di easter_days()

```
echo easter_days (1999);        /* 14, i.e. April 4   */
echo easter_days (1492);        /* 32, i.e. April 22  */
echo easter_days (1913);        /*  2, i.e. March 23  */
```

La data della Pasqua fu definita dal Concilio di Nicea nel 325 d.C. come la Domenica successiva alla prima luna piena dopo l'Equinozio di Primavera. Si assume che l'Equinozio cada sempre il 21 Marzo, quindi il calcolo si riduce alla determinazione della data della luna piena e la data della Domenica seguente. L'algoritmo qui usato fu proposto attorno all'anno 532 d.C. da Dionysius Exiguus (Dionigi il Piccolo). Nel Calendario Giuliano (for years before 1753) un semplice ciclo di 19 anni è usato per traciare le fasi della Luna. Nel Calendario Gregoriano (per gli anni dopo il 1753 - ideato da Clavius e Lilius, e introdotto da Papa Gregorio XIII nell'Ottobre 1582, e in Gran Bretagna

e nelle sue colonie nel Settembre 1752) due fattori correttivi sono aggiunti per rendere più accurato il ciclo.

(Il codice è basato su un programma in C di Simon Kershaw, <webmaster@ely.anglican.org>)

Vedere anche easter_date().

FrenchToJD (PHP 3, PHP 4 >= 4.0.0)

Converte una data del Calendario Repubblicano Francese in un Giorno Giuliano

int **frenchtojd** (int mese, int giorno, int anno) \linebreak

Converte una data del Calendario Repubblicano Francese in un Giorno Giuliano.

Queste funzioni convertono solo le date con gli anni dal 1 al 14 (date Gregoriane dal 22 Settmbre 1792 al 22 Settembre 1806). Questo copre più del periodo in cui fu in uso il calendario.

GregorianToJD (PHP 3, PHP 4 >= 4.0.0)

Converte una data Gregoriana in un Giorno Giuliano

int **gregoriantojd** (int mese, int giorno, int anno) \linebreak

L'intervallo valido per il Calendario Gregoriano è dal 4714 a.C. al 9999 d.C.

Anche se questa funzione può gestire date fino al 4714 a.C., qusto utilizzo potrebbe non avere senso. Il calendario Gregorian fu istituito il 15 Ottobre 1582 (o 5 Ottobre 1582 nel calendario Giuliano). Alcune nazioni non lo accettarono per un lungo periodo. Per esempio, il Regno Unito si convertì nel 1752, L'Unione Sovietica nel in 1918 e la Grecia nel 1923. La maggior parte delle nazioni Europee usavano il calendario Giuliano pdima del Gregoriano.

Esempio 1. Calendar functions

```
<?php
$jd = GregorianToJD (10,11,1970);
echo "$jd\n";
$gregorian = JDToGregorian ($jd);
echo "$gregorian\n";
?>
```

JDDayOfWeek (PHP 3, PHP 4 >= 4.0.0)

Restituisce il giorno della settimana

mixed **jddayofweek** (int giornogiuliano, int modo) \linebreak

Restituisce il giorno della settimana. Può restituire una stringa o un intero a seconda del modo.

Tabella 1. Modi

Modo	Significato
0	Restiutuisce il numero del giorno come intero (0=domenica, 1=lunediì, etc)
1	Restituisce una stringa contenente il giorno della settimana (in inglese-gregoriano)
2	Restituisce una stringa contenente il giorno della settimana abbreviato (in inglese-gregoriano)

JDMonthName (PHP 3, PHP 4 >= 4.0.0)

Restituisce il nome di un mese

string **jdmonthname** (int giornogiuliano, int modo) \linebreak

Restituisce una stringa contenente il nome di un mese. `modo` dice alla funzione verso quale calendario convertire il giorno Giuliano, e che tipo di nome di mese restituire.

Tabella 1. Modi del Calendario

Modo	Significato	Valori
0	Gregoriano abbreviato	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
1	Gregoriano	January, February, March, April, May, June, July, August, September, October, November, December
2	Guliano abbreviato	Jan, Feb, Mar, Apr, May, Jun, Jul, Aug, Sep, Oct, Nov, Dec
3	Guliano	January, February, March, April, May, June, July, August, September, October, November, December
4	Giudeo	Tishri, Heshvan, Kislev, Tevet, Shevat, AdarI, AdarII, Nisan, Iyyar, Sivan, Tammuz, Av, Elul
5	Repubblicano Francese	Vendemiaire, Brumaire, Frimaire, Nivose, Pluviose, Ventose, Germinal, Floreal, Prairial, Messidor, Thermidor, Fructidor, Extra

JDToFrench (PHP 3, PHP 4 >= 4.0.0)

Converte un Giorno Giuliano in una data del Calendario Repubblicano Francese

string **jdtofrench** (int giornogiuliano) \linebreak

Converte un Giorno Giuliano in una data del calendario Repubblicano Francese.

JDToGregorian (PHP 3, PHP 4 >= 4.0.0)

Converte il Giorno Giuliano in data Gregoriana

string **jdtogregorian** (int giornogiuliano) \linebreak

Converte il Giorno Giuliano in una stringa contenente la data Gregoriana nel formato "mese/giorno/anno".

JDToJewish (PHP 3, PHP 4 >= 4.0.0)

Converte un Giorno Giuliano nel Calendario Giudeo

string **jdtojewish** (int giornogiuliano) \linebreak

Converte un Giorno Giuliano nel Calendario Giudeo.

JDToJulian (PHP 3, PHP 4 >= 4.0.0)

Converte un Giorno Giuliano in una data Giuliana

string **jdtojulian** (int giornogiuliano) \linebreak

Converte un Giorno Giuliano in una stringa contenente la data del calendario Giuliano nel formato "mese/giorno/anno".

jdtounix (PHP 4 >= 4.0.0)

Converte un Giorno Giuliano in un timestamp UNIX

int **jdtounix** (int giornogiuliano) \linebreak

Questa funzione restituisce un timestamp UNIX corrispondente al Giorno Giuliano *`giornogiuliano`* o `FALSE` se *`giornogiuliano`* non è all'interno della gamma UNIX (anni Gregoriani tra il 1970 e il 2037 o 2440588 <= *`giornogiuliano`* <= 2465342)

See also unixtojd().

JewishToJD (PHP 3, PHP 4 >= 4.0.0)

Converte una data del Calendario Giudeo in Giorno Giuliano

int **jewishtojd** (int mese, int giorno, int anno) \linebreak

Anche se questa funzione può gestire date fino all'anno 1 (3761 B.C.), questo utilizzo potrebbe non avere senso. Il calendario Giudeo è usato da parecchie migliaia di anni, ma nei primi tempi non c'era una formula per stabilire l'inizio del mese. Il nuovo mese iniziava quando si vedeva la prima volta la luna.

JulianToJD (PHP 3, PHP 4 >= 4.0.0)

Converte una data Giuliana in un Giorno Giuliano

int **juliantojd** (int mese, int giorno, int anno) \linebreak

L'intervallo valido per il Calendario Giuliano è dal 4713 a.C. al 9999 d.C.

Anche se questa funzione può gestire date fino al 4713 a.C., questo utilizzo potrebbe non avere senso. Il calendario fu creato nel 46 a.C., ma i dettagli non furono perfezionati fino almeno al 8 d.D., e forse anche fino al quarto secolo. Inoltre, l'inizio dell'anno variava da una cultura all'altrar - non tutti accettavano Gennaio come primo mese.

Cautela

Il calendario attuale, utilizzato in tutto il mondo, è il calendario Gregoriano. gregoriantojd() può essere utilizzata per convertire queste date nel corrispondente Giorno Giuliano.

unixtojd (PHP 4 >= 4.0.0)

Converte un timestamp UNIX in un Giorno Giuliano

int **unixtojd** ([int timestamp]) \linebreak

Restituisce il Giorno Giuliano di un *`timestamp`* UNIX (secondi dal 1/1/1970), o del giorno corrente se *`timestamp`* non è specificato.

Vedere anche jdtounix().

VII. Funzioni API CCVS

Introduzione

Queste funzioni si interfacciano con le API CCVS, permettendo di lavorare direttamente con CCVS dagli script PHP. CCVS è la soluzione di RedHat (http://www.redhat.com/) per il "mediatore" nella gestione dei pagamenti con carta di credito. Permette di comunicare direttamente con le società di autorizzazione di transazione attraverso una *nix box e un modem. Usando il modulo CCVS per PHP, è possibile procesare direttamente le carte di credito attraverso gli script PHP. Le seguenti informazioni esemplificheranno il processo.

> **Nota:** CCVS è stato abbandonato da Red Hat e non c'è l'intenzione di fornire altre chiavi o contratti di assistenza. Chi cerca un sostituto può considerare MCVE della Main Street Softworks (http://www.mcve.com/) come una possibile alternativa. Il prodotto è simile nella struttura ed ha un supporto documentato per PHP!

Istallazione

Per abilitare il supporto CCVS in PHP, occorre innanzitutto verificare la directory dell'installazione CCVS. Occorrerà poi configurare PHP con l'opzione `--with-ccvs`. Se si usa questa opzione senza specificare il percorso all'installazaione CCVS, il PHP cercherà nel percorso di installazione di CCVS di default (/usr/local/ccvs). Se il CCVS è in un percorso non standard, eseguire configure con: `--with-ccvs=$ccvs_path`, dove $ccvs_path è il percorso dell'installazione di CCVS. Si noti che il supporto CCVS richiede che $ccvs_path/lib e $ccvs_path/include esistano, e include cv_api.h nella directory include e libccvs.a nella directory lib.

Inoltre, un processo ccvsd deve essere attivato con le configurazioni che si vogliono utilizzare negli script PHP. Si dovrà anche assicurarsi che i processi PHP siano eseguiti con lo stesso utente del CCVS (es. se CCVS è installato come utente 'ccvs', i processi PHP devono pure essere eseguiti come utente 'ccvs'.)

Vedere Anche

Informazioni aggiuntive riguardanti CCVS possono essere trovate qui http://www.redhat.com/products/ccvs. RedHat mantiene una documentazione leggermente datata ma utile presso http://www.redhat.com/products/ccvs/support/CCVS3.3docs/ProgPHP.html.

ccvs_add (PHP 4 >= 4.0.2)

Aggiunge dati ad una transazione

string **ccvs_add** (string sessione, string fattura, string tipo_arg, string valore_arg) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_auth (PHP 4 >= 4.0.2)

Esegue un test di autorizzazione al credito su una transazione

string **ccvs_auth** (string sessione, string fattura) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_command (PHP 4 >= 4.0.2)

Esegue un comando caratteristico di un particolare protocollo, quindi non disponibile nelle API di CCVS

string **ccvs_command** (string sessione, string tipo, string val_arg) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_count (PHP 4 >= 4.0.2)

Conta quante transazioni di un dato tipo sono archiviate nel sistema

int **ccvs_count** (string sessione, string tipo) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_delete (PHP 4 >= 4.0.2)

Cancella una transazione

string **ccvs_delete** (string sessione, string fattura) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_done (PHP 4 >= 4.0.2)

Ferma il processo CCVS e ripulisce gli oggetti creati

string **ccvs_done** (string sessione) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_init (PHP 4 >= 4.0.2)

Inizializza CCVS per il successivo utilizzo

string **ccvs_init** (string nome) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_lookup (PHP 4 >= 4.0.2)

Cerca una voce di un determinato tipo nel database #

string **ccvs_lookup** (string sessione, string fattura, int num) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_new (PHP 4 >= 4.0.2)

Crea una nuova transazione, vuota

string **ccvs_new** (string sessione, string fattura) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_report (PHP 4 >= 4.0.2)

Restituisce lo stato del processo di comunicazione

string **ccvs_report** (string sessione, string tipo) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_return (PHP 4 >= 4.0.2)

Trasferisce fondi dal merchant al titolare della carta di credito

string **ccvs_return** (string sessione, string fattura) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_reverse (PHP 4 >= 4.0.2)

Esegue uno storno su un'autorizzazione già processata

string **ccvs_reverse** (string sessione, string fattura) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_sale (PHP 4 >= 4.0.2)

Trasferisce fondi dal titolare della carta di credito al merchant

string **ccvs_sale** (string sessione, string fattura) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_status (PHP 4 >= 4.0.2)

Controlla lo stato di una fattura

string **ccvs_status** (string sessione, string fattura) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_textvalue (PHP 4 >= 4.0.2)

Restuisce il valore testuale reso dalla precedente chiamata di funzione

string **ccvs_textvalue** (string sessione) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ccvs_void (PHP 4 >= 4.0.2)

Esegue uno storno su una transazione già completata

string **ccvs_void** (string sessione, string fattura) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

VIII. Funzioni di supporto COM per Windows

Introduzione

COM è una tecnologia che permette il riutilizzo del codice scritto in un qualsiasi linguaggio utilizzando una convenzione standard di chiamate e nascondendo dietro a delle API i dettagli di implementazione quali la macchina su cui il Componente è conservato e il file eseguibile che lo ospita. Si può pernsare a COM com a un meccanismo avanzato di Remote Procedure Call (RPC) con alcune basi di programmazione ad oggetti. COM separa l'implementazione dall'interfaccia.

COM incoraggia il versioning, la separazione tra interfaccia e implementazione e l'occultamento dei dettagli dell'implementazione, come la posizione dell'eseguibile e il linguaggio di programmazione.

Le funzioni COM sono disponibili solo sulle versioni Windows di PHP.

Costanti Predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata

compilata nel PHP o se è stata caricata dinamicamente a runtime.

`CLSCTX_INPROC_SERVER` (integer)

`CLSCTX_INPROC_HANDLER` (integer)

`CLSCTX_LOCAL_SERVER` (integer)

`CLSCTX_REMOTE_SERVER` (integer)

`CLSCTX_SERVER` (integer)

`CLSCTX_ALL` (integer)

`VT_NULL` (integer)

`VT_EMPTY` (integer)

`VT_UI1` (integer)

`VT_I2` (integer)

`VT_I4` (integer)

`VT_R4` (integer)

`VT_R8` (integer)

`VT_BOOL` (integer)

`VT_ERROR` (integer)

`VT_CY` (integer)

`VT_DATE` (integer)

`VT_BSTR` (integer)

`VT_DECIMAL` (integer)

(http://www.microsoft.com/Com/resources/comdocs.asp) o si guardi anche la documentazione di Don Box su Yet Another COM Library (YACL) (http://www.developmentor.com/dbox/yacl.htm)

COM (unknown)

classe COM

Sinossi

```
$obj = new COM("server.object")
```

La classe COM fornisce un ambiente per integrare (D)COM components negli script php.

string **COM::COM** (string nome_modulo [, string nome_server [, int codepage]]) \linebreak

costruttore della classe COM. Parametri:

nome_modulo

nome o class-id del componente desiderato.

nome_server

nome del server DCOM dal quale deve essere richiamato il componente. Se `NULL`, si assume `localhost`. Per permettere l'uso di DCOM il parametro `com.allow_dcom` deve essere impostato a `TRUE` in `php.ini`.

codepage

specifica la codepage che verrà usata per convertire le stringhe di PHP in stringhe Unicode e viceversa. I valori possibili sono `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` e `CP_UTF8`.

Esempio 1. esempio di COM (1)

```
// esecuzione di Word
$word = new COM("word.application") or die("Non sono riuscito ad eseguire Word");
print "Word caricato, versione {$word->Version}\n";

//lo porta in primo piano
$word->Visible = 1;

//apre un documento vuoto
$word->Documents->Add();

//esegue un po' di operazioni inutili
$word->Selection->TypeText("Questa è una prova...");
$word->Documents[1]->SaveAs("Prova inutile.doc");

//chiude Word
$word->Quit();

//libera l'oggetto
$word->Release();
```

```
$word = null;
```

Esempio 2. esempio di COM (2)

```
$conn = new COM("ADODB.Connection") or die("non riesco ad attivare ADO");
$conn->Open("Provider=SQLOLEDB; Data Source=localhost;
Initial Catalog=database; User ID=user; Password=password");

$rs = $conn->Execute("SELECT * FROM unatabella");    // Recordset

$num_colonne = $rs->Fields->Count();
echo $num_columns . "\n";

for ($i=0; $i < $num_colonne; $i++)
{
    $campi[$i] = $rs->Fields($i);
}

$contorighe = 0;
while (!$rs->EOF)
{
    for ($i=0; $i < $num_colonne; $i++)
    {
        echo $campi[$i]->value . "\t";
    }
    echo "\n";
    $contorighe++;            // incrementa contorighe
    $rs->MoveNext();
}

$rs->Close();
$conn->Close();

$rs->Release();
$conn->Release();

$rs = null;
$conn = null;
```

VARIANT (unknown)

classe VARIANT

Sinossi

```
$vVar = new VARIANT($var)
```

Un semplice contenitore per includere le variabili in strutture VARIANT.

string **VARIANT::VARIANT** ([mixed valore [, int tipo [, int codepage]]]) \linebreak

costruttore della classe VARIANT. Parametri:

valore

valore iniziale. se omesso viene creato un oggetto VT_EMPTY.

tipo

specifica il tipo di contenuto dell'oggetto VARIANT. I valori possibili sono `VT_UI1`, `VT_UI2`, `VT_UI4`, `VT_I1`, `VT_I2`, `VT_I4`, `VT_R4`, `VT_R8`, `VT_INT`, `VT_UINT`, `VT_BOOL`, `VT_ERROR`, `VT_CY`, `VT_DATE`, `VT_BSTR`, `VT_DECIMAL`, `VT_UNKNOWN`, `VT_DISPATCH` and `VT_VARIANT`. Questi valori sono mutuallmente esclusivi, ma possono essere combinati con `VT_BYREF` to specify being a value. If omitted, the type of *`value`* is used. Consult the msdn library for additional information.

codepage

specifica la codepage che è usata per convertire le stringhe PHP in stringhe unicode e viceversa. I valori possibili sono `CP_ACP`, `CP_MACCP`, `CP_OEMCP`, `CP_SYMBOL`, `CP_THREAD_ACP`, `CP_UTF7` e `CP_UTF8`.

com_addref (PHP 4 >= 4.1.0)

Incrementa il contatore di referenze.

void **com_addref** (void) \linebreak

Incrementa il contatore di referenze.

com_get (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Ricava il valore di una proprietà di un componente COM

mixed **com_get** (resource oggetto_com, string nome_prop) \linebreak

Restituisce il valore della proprietà *`nome_prop`* del componente COM referenziato da *`oggetto_com`*. Restituisce `FALSE` in caso di errore.

com_invoke (PHP 3>= 3.0.3)

Chiama un metodo di un componente COM.

mixed **com_invoke** (resource oggetto_com, string nome_funzione [, mixed parametri, ...]) \linebreak

com_invoke() chiama un metodo del componente COM referenziato da *oggetto_com*. Restituisce FALSE in caso di errore, altrimenti restituisce il valore di ritorno di *nome_funzione*.

com_isenum (PHP 4 >= 4.1.0)

Recupera un IEnumVariant

void **com_isenum** (object modulo_com) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

com_load (PHP 3>= 3.0.3)

Crea un nuovo riferimento a un componente COM

string **com_load** (string nome_modulo [, string nome_server [, int codepage]]) \linebreak

com_load() crea un nuovocomponente COM e ne restituisce un riferimento. Restituisce FALSE in caso di errore. I possibili valori per *codepage* sono CP_ACP, CP_MACCP, CP_OEMCP, CP_SYMBOL, CP_THREAD_ACP, CP_UTF7 e CP_UTF8.

com_load_typelib (PHP 4 >= 4.1.0)

Carica una Typelib

void **com_load_typelib** (string nome_typelib [, int ignora_maiuscole]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

com_propget (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Ricava il valore di una proprietà di un componente COM

mixed **com_propget** (resource oggetto_com, string nome_prop) \linebreak

Questa funzione è un alias di com_get().

com_propput (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Assegna un valore a una proprietà di un oggetto COM

void **com_propput** (resource oggetto_com, string nome_prop, mixed valore) \linebreak

Questa funzione è un alias di com_set().

com_propset (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Assegna un valore a una proprietà di un oggetto COM

void **com_propset** (resource oggetto_com, string nome_prop, mixed valore) \linebreak

Questa funzione è un alias di com_set().

com_release (PHP 4 >= 4.1.0)

Decrementa il contaotre di referenze.

void **com_release** (void) \linebreak

Decrementa il contatore di referenze.

com_set (PHP 3>= 3.0.3, PHP 4 >= 4.0.5)

Assegna un valore a una proprietà di un oggetto COM

void **com_set** (resource oggetto_com, string nome_prop, mixed valore) \linebreak

Imposta il valore della proprietà *nome_prop* del componente COM referenziato da *oggetto_com*. Restituisce il valore appena impostato in caso di successo,, FALSE altrimenti.

IX. Funzioni per Classi/Oggetti

Introduzione

Queste funzioni permettono di ottenere informazioni sulle classi e sulle istanze degli oggetti. Si può ricavare il nome della classe da cui deriva un dato oggetto, come le sue proprietà e i suoi metodi. Utilizzando queste funzioni si ottiene, non solo a quale classe appartiene un dato oggetto, ma anche i suoi "padri" (ad esempio da quale classe è derivata la classe dell'oggetto).

Esempi

In questo esempio, prima si definisce una classe base, quindi una seconda che deriva dalla prima. La classe base descrive gli aspetti generali degli ortaggi, se è commestibile e quale sia il colore. La classe derivata `Spinaci` aggiunge i metodi di cottura e di verifica della completa cottura.

Esempio 1. classi.inc

```
<?php

// classe base con proprietà e metodi
class Ortaggio {

    var $commestibile;
    var $colore;

    function Ortaggio( $commestibile, $colore="verde" ) {
        $this->commestibile = $commestibile;
        $this->colore = $colore;
    }

    function e_commestibile() {
        return $this->commestibile;
    }

    function che_colore_ha() {
        return $this->colore;
    }

} // Fine della classe ortaggio

// Estensione della classe base
class Spinaci extends Ortaggio {

    var $cotto = false;

    function Spinaci() {
        $this->Ortaggio( true, "verde" );
    }

    function cuocilo() {
```

```
        $this->cotto = true;
    }

    function e_cotto() {
        return $this->cotto;
    }

} // Fine della classe spinaci

?>
```

A questo punto si istanziano 2 oggetti a partire da queste classi e si visualizzeranno le informazioni relative a questi oggetti, compresi i loro padri. Verranno anche inserite funzioni di utilità principalmente con lo scopo di rendere chiara la visualizzazione delle variabili.

Esempio 2. test_script.php

```
<pre>
<?php

include "classi.inc";

// Funzioni di utilità

function visualizza_var($oggetto) {
    $matrice = get_object_vars($oggetto);
    while (list($prop, $val) = each($matrice))
        echo "\t$prop = $val\n";
}

function visualizza_metodi($oggetto) {
    $matrice = get_class_methods(get_class($oggetto));
    foreach ($matrice as $metodo)
        echo "\tfunzione $metodo()\n";
}

function padri_classe($oggetto, $classe) {
    global $$oggetto;
    if (is_subclass_of($$oggetto, $classe)) {
        echo "Oggetto $oggetto appartiene alla classe ".get_class($$oggetto);
        echo " derivata da $classe\n";
    } else {
        echo "Oggetto $oggetto non deriva da una sottoclasse di $classe\n";
    }
}

// Istanzia 2 oggetti

$pomodoro = new Ortaggio(true,"rosso");
$frondoso = new Spinaci();

// Visualizza le informazioni sugli oggetti
```

```
echo "pomodoro: CLASSE ".get_class($pomodoro)."\n";
echo "frondoso: CLASSE ".get_class($frondoso);
echo ", PADRE ".get_parent_class($frondoso)."\n";

// visualizza le proprietà di pomodoro
echo "\npomodoro: Proprietà\n";
visualizza_var($pomodoro);

// e i metodi di frondoso
echo "\nfrondoso: Metodi\n";
visualizza_metodi($frondoso);

echo "\nPadri:\n";
padri_classe("frondoso", "Spinaci");
padri_classe("frondoso", "Ortaggio");
?>
</pre>
```

Un aspetto da notare nell'esempio precedente è che l'oggetto `$frondoso` è un'istanza della classe `Spinaci` che a sua volta è una sottoclasse di `Ortaggio`, quindi l'ultima parte dell'esempio visualizzerà:

```
       [...]
Padri:
Oggetto frondoso non deriva da una sottoclasse di Spinaci
Oggetto frondoso appartiene alla classe spinaci derivata da Ortaggio
```

call_user_method (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Chiama un metodo dell'oggetto indicato [deprecated]

mixed **call_user_method** (string nome_metodo, object oggetto [, mixed parametro [, mixed ...]]) \linebreak

> **Attenzione**
>
> A partire dalla versione 4.1.0 l'uso della funzione **call_user_method()** è sconsigliato; in sostituzione utilizzare la serie call_user_func() con la sintassi `array(&$obj, "method_name")`.

Richiama il metodo indicato da *`nome_metodo`* dell'oggetto *`oggetto`*. Di seguito si fornisce un esempio di utilizzo. Qui si definisce una classe, si istanzia un oggetto, e si utilizza **call_user_method()** per richiamare il metodo `stampa_info`

```
<?php
class Stato {
    var $NOME;
    var $TLD;

    function Stato($nome, $tld) {
        $this->NOME = $nome;
        $this->TLD = $tld;
    }

    function stampa_info($prestr="") {
        echo $prestr."Stato: ".$this->NOME."\n";
        echo $prestr."Dominio di primo livello: ".$this->TLD."\n";
    }
}

$paese = new Stato("Peru","pe");

echo "* Richiamo il metodo direttamente\n";
$paese->stampa_info();

echo "\n* utilizzo dello stesso metodo in modo indiretto\n";
call_user_method("stampa_info", $paese, "\t");
?>
```

Vedere anche call_user_func_array(), call_user_func(), call_user_method_array().

call_user_method_array (PHP 4 >= 4.0.5)

Richiama il metodo dato con un array di parametri [deprecated]

mixed **call_user_method_array** (string nome_metodo, object oggetto [, array array_parametri]) \linebreak

> **Attenzione**
>
> A partire dalla versione 4.1.0 è sconsigliato l'uso di **call_user_method_array()**; in sostituzione utilizzare la serie di funzioni call_user_func_array() con sintassi `array(&$obj, "method_name")`.

Richiama il metodo indicato da *`nome_metodo`* dell'oggetto *`oggetto`*, utilizzando i parametri forniti in *`array_parametri`*.

Vedere anche: call_user_func_array(), call_user_func(), call_user_method().

> **Nota:** Questa funzione è stata aggiunta al codice CVS dopo la release 4.0.4pl1 di PHP

class_exists (PHP 4 >= 4.0.0)

Verifica se una classe è stata definita

bool **class_exists** (string nome_classe) \linebreak

Questa classe restituisce `TRUE` se la classe indicata dal parametro *`nome_classe`* è stata definita, altrimenti restituisce `FALSE` .

get_class (PHP 4 >= 4.0.0)

Restituisce il nome della classe di un oggetto

string **get_class** (object oggetto) \linebreak

Questa funzione restituisce il nome della classe di cui l'oggetto *`oggetto`* è un'istanza. Restituisce `FALSE` se *`oggetto`* non è un oggetto.

> **Nota: get_class()** restituisce il nome delle classi definite dagli utenti in minuscolo. Viceversa le classi definite nelle estensioni di PHP sono restituite nella notazione originale.

Vedere anche get_parent_class(), gettype() e is_subclass_of()

get_class_methods (PHP 4 >= 4.0.0)

Restituisce un array con i nomi dei metodi della classe

array **get_class_methods** (string nome_classe) \linebreak

Questa funzione restituisce un array contenente i nomi dei metodi definiti per la classe specificata da *nome_classe*.

> **Nota:** Dalla versione 4.0.6 di PHP, si può specificare direttamente l'oggetto anziché la classe nel parametro *nome_classe*. Ad esempio:

```
$metodi_della_classe = get_class_methods($mia_classe); // vedere di seguito l'esempio completo
```

Esempio 1. Esempio di get_class_methods()

```
<?php

class miaclasse {
    // costruttore
    function miaclasse() {
        return(true);
    }

    // metodo 1
    function funzione1() {
        return(true);
    }

    // metodo 2
    function funzione2() {
        return(true);
    }
}

$mio_oggetto = new miaclasse();

$metodi = get_class_methods(get_class($mio_oggetto));

foreach ($metodi as $nome_metodo) {
    echo "$nome_metodo\n";
}

?>
```

Produrrà:

```
miaclasse
funzione1
```

```
funzione2
```

Vedere anche get_class_vars() e get_object_vars()

get_class_vars (PHP 4 >= 4.0.0)

Restituisce un array con le proprietà di default della classe

array **get_class_vars** (string nome_classe) \linebreak

Questa funzione restituisce un array associativo contenente le proprietà di default della classe. Gli elementi dell'array prodotto sono nel formato *`nomevariabile => valore`*.

> **Nota:** Le variabili della classe non inizializzate non sono elencate da **get_class_vars()**.

Esempio 1. get_class_vars() esempio

```
<?php

class miaclasse {

    var $var1; // questa variabile non ha un valore di default...
    var $var2 = "xyz";
    var $var3 = 100;

    // costruttore
    function miaclasse() {
        return(true);
    }

}

$mia_classe = new miaclasse();

$variabili = get_class_vars(get_class($mia_classe));

foreach ($variabili as $nome => $valore) {
    echo "$nome : $valore\n";
}

?>
```

Produrrà:

```
var2 : xyz
var3 : 100
```

Vedere anche get_class_methods() e get_object_vars()

get_declared_classes (PHP 4 >= 4.0.0)

Restituisce un array con il nome delle classi definite

array **get_declared_classes** (void) \linebreak

Questa funzione restituisce un array con i nomi delle classi definite all'interno dello script corrente.

> **Nota:** Nella versione 4.0.1pl2 di PHP, in testa all'array erano indicate tre ulteriori classi: `stdClass` (definita in `Zend/zend.c`), `OverloadedTestClass` (definita in `ext/standard/basic_functions.c`) e `Directory` (definita in `ext/standard/dir.c`).
>
> Occorre notare che, in base a quali librerie sono state compilate in PHP, possono essere rilevate ulteriori classi. Questo significa, anche, che non si potranno definire delle classi con questi nomi. Un'elenco delle classi predefinite è nella sezione Predefined Classes dell'appendice.

get_object_vars (PHP 4 >= 4.0.0)

Restituisce un array associativo con le proprietà dell'oggetto

array **get_object_vars** (object oggetto) \linebreak

Questa funzione restituisce un array associativo con le proprietà definite nell'oggetto passato nel parametro *`oggetto`* . Le variabili, dichiarate nella classe di cui *`oggetto`* è un'istanza, a cui non sia stato ancora assegnato un valore, non saranno restituite nell'array.

Esempio 1. Utilizzo di get_object_vars()

```
<?php
class Point2D {
    var $x, $y;
    var $etichetta;

    function Point2D($x, $y) {
        $this->x = $x;
        $this->y = $y;
    }

    function setetichetta($etichetta) {
        $this->etichetta = $etichetta;
```

```
    }

    function getPoint() {
        return array("x" => $this->x,
                     "y" => $this->y,
                     "etichetta" => $this->etichetta);
    }
}

// "$etichetta" è dichiarata ma non definita
$p1 = new Point2D(1.233, 3.445);
print_r(get_object_vars($p1));

$p1->setetichetta("point #1");
print_r(get_object_vars($p1));

?>
```

L'output del programma precedente sarà:

```
Array
(
     [x] => 1.233
     [y] => 3.445
)

Array
(
     [x] => 1.233
     [y] => 3.445
     [label] => point #1
)
```

Vedere anche get_class_methods() e get_class_vars()

get_parent_class (PHP 4 >= 4.0.0)

Restituisce il nome della classe genitrice di un oggetto o di una classe

string **get_parent_class** (mixed oggetto) \linebreak

Se `oggetto` è un oggetto, la funzione restituisce il nome del genitore della classe di cui `oggetto` è un'istanza.

Se `oggetto` è una stringa, la funzione restituisce il nome della classe genitrice della classe di cui `oggetto` indica il nome. Questa funzionalità è stata aggiunta nella versione 4.0.5 di PHP.

Vedere anche get_class(), is_subclass_of()

is_a (PHP 4 >= 4.2.0)

Restituisce vero se l'oggetto appartiene a questa classe o se ha questa classe tra i suoi genitori

bool **is_a** (object object, string class_name) \linebreak

Questa funzione restituisce `TRUE` appartiene a questa classe oppure ha questa classe tra i suoi genitori, `FALSE` in caso diverso.

Vedere anche get_class(), get_parent_class() e is_subclass_of().

is_subclass_of (PHP 4 >= 4.0.0)

Restituisce vero se l'oggetto ha questa classe come uno dei suoi genitori

bool **is_subclass_of** (object oggetto, string nome_classe) \linebreak

Questa funzione restituisce `TRUE` se *`obj`*, appartiene ad una sottoclasse di *`nome_classe`*, altrimenti `FALSE`.

Vedere anche get_class(), get_parent_class() e is_a().

method_exists (PHP 4 >= 4.0.0)

Verifica se il metodo esiste nella classe

bool **method_exists** (object object, string nome_metodo) \linebreak

Questa funzione restituisce `TRUE` se il metodo indicato dal parametro *`nome_metodo`* è stato nell'oggetto indicato da *`oggetto`*, altrimenti `FALSE`.

X. Funzioni ClibPDF

ClibPDF permette di creare documenti PDF utilizzando PHP. E’ disponibile per il download a FastIO (http://www.fastio.com/), ma richiede che si acquisti una licenza per l’utilizzo commerciale. La funzionalità e l’ API di ClibPDF sono simili a quelle di PDFlib.

Questa documentazione dovrebbe essere letta accanto al manuale di ClibPDF dato che quest’ultimo spiega la libreria più nel dettaglio.

Molte funzioni nella ClibPDF nativa e nel modulo PHP , così come nella PDFlib hanno lo stesso nome. Tutte le funzioni ad eccezione di cpdf_open() hanno l’identificatore del documento come loro primo parametro.

Attualmente questo identificatore non è usato internamente dato che ClibPDF non supporta la creazione di svariati documenti PDF contemporaneamente. Attualmente non bisognerebbe provare a farlo, dato che i risultati sono imprevedibili. Non è possibile controllare quali siano le conseguenze in un ambiente multi processo. Secondo l’autore di ClibPDF questo cambierà in una delle prossime release (la versione corrente quando questo è stato scritto è la 1.10). Se si ha bisogno di questa funzionalità utilizzare il modulo pdflib.

> **Nota:** La funzione cpdf_set_font() è cambiata dal PHP 3 per supportare i font asiatici. Il parametro per l’encoding non è più un numero intero, ma una stringa.

Un’interessante caratteristica di ClibPDF (e PDFlib) è l’abilità di creare il documento PDF completamente in memoria senza utilizzare file temporanei. Inoltre fornisce l’abilità di passare le coordinare in un’unità di lunghezza predefinita. (Questa caratteristica puo’essere simulata da pdf_translate() quando si utilizzano le funzioni PDFlib.)

Un’altra interessante caratteristica di ClibPDF è il fatto che ogni pagina può essere modificata in qualsiasi momento anche se è già stata aperta una nuova pagina. La funzione cpdf_set_current_page() permette di lasciare la pagina corrente e modificare un’altra pagina.

La maggioranza delle funzioni sono abbastanza facili da usare. La parte più difficile è probabilmente creare un simplice documento PDF. L’esempio seguente dovrebbe essere utile per cominciare. Crea un documento con una pagina. La pagina contiene il testo "Times-Roman" con un font di 30pt. Il testo è sottolineato.

Esempio 1. Semplice esempio ClibPDF

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842, 1.0);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
cpdf_begin_text($cpdf);
cpdf_set_font($cpdf, "Times-Roman", 30, "WinAnsiEncoding");
cpdf_set_text_rendering($cpdf, 1);
cpdf_text($cpdf, "Times Roman outlined", 50, 750);
cpdf_end_text($cpdf);
cpdf_moveto($cpdf, 50, 740);
cpdf_lineto($cpdf, 330, 740);
cpdf_stroke($cpdf);
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
```

```
cpdf_close($cpdf);
?>
```

La distribuzione pdflib contiene un esempio più complicato che crea una serie di pagine con un orologio analogico. Di seguito si trova questo esempio convertito in PHP utilizzando l’estensione ClibPDF:

Esempio 2. esempio pdfclock dalla distribuzione pdflib 2.0

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 40;

$pdf = cpdf_open(0);
cpdf_set_creator($pdf, "pdf_clock.php3");
cpdf_set_title($pdf, "Analog Clock");

while($pagecount-- > 0) {
  cpdf_page_init($pdf, $pagecount+1, 0, 2 * ($radius + $margin), 2 * ($radius + $mar-
gin), 1.0);

  cpdf_set_page_animation($pdf, 4, 0.5, 0, 0, 0);  /* wipe */

  cpdf_translate($pdf, $radius + $margin, $radius + $margin);
  cpdf_save($pdf);
  cpdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);

  /* linee dei minuti */
  cpdf_setlinewidth($pdf, 2.0);
  for ($alpha = 0; $alpha &lt; 360; $alpha += 6)
    {
    cpdf_rotate($pdf, 6.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin/3, 0.0);
    cpdf_stroke($pdf);
    }

  cpdf_restore($pdf);
  cpdf_save($pdf);

  /* linee dei 5 minuti */
  cpdf_setlinewidth($pdf, 3.0);
  for ($alpha = 0; $alpha &lt; 360; $alpha += 30)
  {
    cpdf_rotate($pdf, 30.0);
    cpdf_moveto($pdf, $radius, 0.0);
    cpdf_lineto($pdf, $radius-$margin, 0.0);
    cpdf_stroke($pdf);
  }

  $ltime = getdate();

  /* disegna la lancetta delle ore */
```

```
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['minutes']/60.0) + $ltime['hours'] - 3.0) * 30.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius/2, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* disegna la lancetta dei minuti */
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds']/60.0) + $ltime['minutes'] - 15.0) * 6.0);
  cpdf_moveto($pdf, -$radius/10, -$radius/20);
  cpdf_lineto($pdf, $radius * 0.8, 0.0);
  cpdf_lineto($pdf, -$radius/10, $radius/20);
  cpdf_closepath($pdf);
  cpdf_fill($pdf);
  cpdf_restore($pdf);

  /* disegna la lancetta dei secondi */
  cpdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
  cpdf_setlinewidth($pdf, 2);
  cpdf_save($pdf);
  cpdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
  cpdf_moveto($pdf, -$radius/5, 0.0);
  cpdf_lineto($pdf, $radius, 0.0);
  cpdf_stroke($pdf);
  cpdf_restore($pdf);

  /* disegna un piccolo cerchio al centro */
  cpdf_circle($pdf, 0, 0, $radius/30);
  cpdf_fill($pdf);

  cpdf_restore($pdf);

  cpdf_finalize_page($pdf, $pagecount+1);
}

cpdf_finalize($pdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($pdf);
cpdf_close($pdf);
?>
```

cpdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Aggiunge un'annotazione

void **cpdf_add_annotation** (int documento pdf, float llx, float lly, float urx, float ury, string titolo, string contenuto [, int modo]) \linebreak

La **cpdf_add_annotation()** aggiunge una nota con l'angolo in basso a sinistra in (`llx`, `lly`) e l'angolo in alto a destra in (`urx`, `ury`).

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

cpdf_add_outline (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Aggiunge un segnalibro per la pagina corrente

void **cpdf_add_outline** (int documento pdf, string testo) \linebreak

La funzione **cpdf_add_outline()** aggiunge un segnalibro con il testo `testo` che punta alla pagina corrente.

Esempio 1. Aggiungere il contorno alla pagina

```
<?php
$cpdf = cpdf_open(0);
cpdf_page_init($cpdf, 1, 0, 595, 842);
cpdf_add_outline($cpdf, 0, 0, 0, 1, "Page 1");
// ...
// alcuni disegni
// ...
cpdf_finalize($cpdf);
Header("Content-type: application/pdf");
cpdf_output_buffer($cpdf);
cpdf_close($cpdf);
?>
```

cpdf_arc (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Disegna un arco

void **cpdf_arc** (int documento pdf, float x-coor, float y-coor, float raggio, float inizio, float fine [, int modo]) \linebreak

La funzione **cpdf_arc()** disegna un arco con centro nel punto (`x-coor`, `y-coor`) e raggio `radius`, che comincia all'angolo `inizio` e che termina all'angolo `fine`.

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_circle().

cpdf_begin_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Avvia una sezione di testo

void **cpdf_begin_text** (int documento pdf) \linebreak

La funzione **cpdf_begin_text()** avvia una sezione di testo. Deve essere terminata con cpdf_end_text().

Esempio 1. Output di testo

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Un p&ograve; di testo");
cpdf_end_text($pdf)
?>
```

Vedere anche cpdf_end_text().

cpdf_circle (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Disegna un cerchio

void **cpdf_circle** (int documento pdf, float x-coor, float y-coor, float raggio [, int modo]) \linebreak

La funzione **cpdf_circle()** disegna un cerchio con centro nel punto (*x-coor*, *y-coor*) e raggio *raggio*.

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_arc().

cpdf_clip (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Taglia dal path corrente

void **cpdf_clip** (int documento pdf) \linebreak

La funzione **cpdf_clip()** taglia tutti i disegni dal path corrente.

cpdf_close (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Chiude il documento PDF

void **cpdf_close** (int documento pdf) \linebreak

La funzione **cpdf_close()** chiude il documento pdf. Questa dovrebbe essere l'ultima funzione dopo persino cpdf_finalize(), cpdf_output_buffer() e cpdf_save_to_file().

Vedere anche cpdf_open().

cpdf_closepath (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Chiude la path corrente

void **cpdf_closepath** (int documento pdf) \linebreak

La funzione **cpdf_closepath()** chiude la path corrente.

cpdf_closepath_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Close, fill and stroke current path

void **cpdf_closepath_fill_stroke** (int pdf document) \linebreak

The **cpdf_closepath_fill_stroke()** function closes, fills the interior of the current path with the current fill color and draws current path.

See also cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_closepath_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Chiude il path e disegna una linea lungo il path

void **cpdf_closepath_stroke** (int documento pdf) \linebreak

La funzione **cpdf_closepath_stroke()** è una combinazione di cpdf_closepath() e cpdf_stroke(). Dopo libera il path.

Vedere anche cpdf_closepath() e cpdf_stroke().

cpdf_continue_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Scrive del testo nella riga seguente

void **cpdf_continue_text** (int documento pdf, string testo) \linebreak

La funzione **cpdf_continue_text()** scrive la stringa `testo` nella riga seguente.

Vedere anche cpdf_show_xy(), cpdf_text(), cpdf_set_leading() e cpdf_set_text_pos().

cpdf_curveto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Disegna una curva

void **cpdf_curveto** (int documento pdf, float x1, float y1, float x2, float y2, float x3, float y3 [, int modo]) \linebreak

La funzione **cpdf_curveto()** disegna una curva di Bezier dal punto corrente al punto (`x3`, `y3`) utilizzando (`x1`, `y1`) e (`x2`, `y2`) come punti di controllo.

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_moveto(), cpdf_rmoveto(), cpdf_rlineto() e cpdf_lineto().

cpdf_end_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Termina la sezione di testo

void **cpdf_end_text** (int documento pdf) \linebreak

La funzione **cpdf_end_text()** termina una sezione di testo cominciata con cpdf_begin_text().

Esempio 1. Output di testo

```
<?php
cpdf_begin_text($pdf);
cpdf_set_font($pdf, 16, "Helvetica", "WinAnsiEncoding");
cpdf_text($pdf, 100, 100, "Un p&ograve; di testo");
cpdf_end_text($pdf)
?>
```

Vedere anche cpdf_begin_text().

cpdf_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Riempie il path corrente

void **cpdf_fill** (int documento pdf) \linebreak

La funzione **cpdf_fill()** riempie l'interno del path corrente con il colore di riempimento corrente.

Vedere anche cpdf_closepath(), cpdf_stroke(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill() e cpdf_setrgbcolor().

cpdf_fill_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fill and stroke current path

void **cpdf_fill_stroke** (int pdf document) \linebreak

The **cpdf_fill_stroke()** function fills the interior of the current path with the current fill color and draws current path.

See also cpdf_closepath(), cpdf_stroke(), cpdf_fill(), cpdf_setgray_fill(), cpdf_setgray(), cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_finalize (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Chiude il documento

void **cpdf_finalize** (int documento pdf) \linebreak

La funzione **cpdf_finalize()** chiude il documento. Bisogna comunque richiamare cpdf_close()

Vedere anche cpdf_close().

cpdf_finalize_page (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Chiude la pagina

void **cpdf_finalize_page** (int documento pdf, int numero della pagina) \linebreak

La funzione **cpdf_finalize_page()** chiude la pagina numero *`numero della pagina`*.

Questa funzione serve solo a preservare la memoria. Una pagina chiusa occupa meno memoria ma non può più essere modificata.

Vedere anche cpdf_page_init().

cpdf_global_set_document_limits (PHP 4 >= 4.0.0)

Sets document limits for any pdf document

void **cpdf_global_set_document_limits** (int maxpages, int maxfonts, int maximages, int maxannotations, int maxobjects) \linebreak

The **cpdf_global_set_document_limits()** function sets several document limits. This function has to be called before cpdf_open() to take effect. It sets the limits for any document open afterwards.

See also cpdf_open().

cpdf_import_jpeg (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Apre un'immagine JPEG

int **cpdf_import_jpeg** (int documento pdf, string nome del file, float x-coor, float y-coor, float angolo, float larghezza, float altezza, float scala-x, float scala-y [, int modo]) \linebreak

La funzione **cpdf_import_jpeg()** apre un'immagine contenuta nel file con nome *nome del file*. Il formato dell'immagine deve essere jpeg. L' immagine viene inserita nella pagina corrente alla posizione (*x-coor*, *y-coor*). L'immagine è ruotata di *angolo* gradi.

Il parametro facoltativo *modo* determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_place_inline_image().

cpdf_lineto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Disegna una linea

void **cpdf_lineto** (int documento pdf, float x-coor, float y-coor [, int modo]) \linebreak

La funzione **cpdf_lineto()** disegna una linea dal punto corrente al punto con coordinate (*x-coor*, *y-coor*).

Il parametro facoltativo *modo* determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_moveto(), cpdf_rmoveto() e cpdf_curveto().

cpdf_moveto (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setta il punto corrente

void **cpdf_moveto** (int documento pdf, float x-coor, float y-coor [, int modo]) \linebreak

La funzione **cpdf_moveto()** setta il punto corrente alle coordinate *x-coor* e *y-coor*.

Il parametro facoltativo *modo* determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

cpdf_newpath (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Inizia un nuovo path

void **cpdf_newpath** (int documento pdf) \linebreak

La funzione **cpdf_newpath()** inizia un nuovo path sul documento indicato dal parametro *documento pdf*.

cpdf_open (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Apre un nuovo documento pdf

int **cpdf_open** (int compressione [, string nome file]) \linebreak

La funzione **cpdf_open()** apre un nuovo documento pdf. Il primo parametro setta il fattore di compressione del documento se è diverso da 0. Il secondo parametro, opzionale, setta il file in cui il documento viene scritto. Se è omesso il documento è creato in memoria e può essere scritto su un file con cpdf_save_to_file() oppure scritto sullo standard output con cpdf_output_buffer().

> **Nota:** Il valore di ritorno sarà necessario nelle future versione di ClibPDF come primo parametro di tutte le altre funzioni che scrivono sul documento PDF.
>
> La libreria ClibPDF considera il nome del file "-" come sinonimo di stdout. Se il PHP è compilato come modulo di apache questo però non funziona perchè il modo in cui ClibPDF scrive sullo stdout non funziona con apache. Si può risolvere questo problema saltando il nome del file e utilizzando cpdf_output_buffer() per scrivere il documento pdf.

Vedere anche cpdf_close() e cpdf_output_buffer().

cpdf_output_buffer (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Scrive il documento pdf nel buffer di memoria

void **cpdf_output_buffer** (int documento pdf) \linebreak

La funzione **cpdf_output_buffer()** scrive il documento pdf nello stdout. Il documento deve essere creato in memoria, cosa che si verifica se cpdf_open() è richiamata senza il nome di un file come parametro.

Vedere anche cpdf_open().

cpdf_page_init (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Inizia una nuova pagina

void **cpdf_page_init** (int documento pdf, int numero di pagina, int orientamento, float altezza, float larghezza [, float unità]) \linebreak

La funzione **cpdf_page_init()** inizia una nuova pagina con altezza `height` e larghezza `width`. La pagina ha numero `numero di pagina` e orientamento `orientamento`. `orientamento` può essere 0 per l'orientamento verticale e 1 per l'orizzontale. L'ultimo parametro facoltativo `unità` setta le unità per il sistema di coordinate. Il valore deve essere il numero di punti postscript per unità. Dato che un pollice corrisponde a 72 punti, un valore di 72 setterà l'unità ad un pollice. Il valore di default è sempre 72.

Vedere cpdf_set_current_page().

cpdf_place_inline_image (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Inserisce un'immagine nella pagina

void **cpdf_place_inline_image** (int documento pdf, int immagine, float x-coor, float y-coor, float angolo, float larghezza, float altezza [, int modo]) \linebreak

La funzione **cpdf_place_inline_image()** inserisce un'immagine creata con le funzioni per le immagini di PHP nella pagina alla posizione (`x-coor`, `y-coor`). L'immagine può essere ridimensionata nello stesso tempo.

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_import_jpeg().

cpdf_rect (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Disegna un rettangolo

void **cpdf_rect** (int documento pdf, float x-coor, float y-coor, float larghezza, float altezza [, int modo]) \linebreak

La funzione **cpdf_rect()** disegna un rettangolo con l'angolo in basso a sinistra nel punto (`x-coor`, `y-coor`). La larghezza è settata a `larghezza`. L'altezza è settata a `altezza`.

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

cpdf_restore (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Ristabilisce le impostazioni salvate in precedenza

void **cpdf_restore** (int documento pdf) \linebreak

La funzione **cpdf_restore()** ristabilisce le impostazioni salvate con cpdf_save(). Funziona come il comando postscript grestore. Molto utile se si vuole traslare o ruotare un oggetto senza intaccare gli altri oggetti.

Esempio 1. Salvare/Richiamare

```
<?php
cpdf_save($pdf);
// effettuare qualsiasi tipo di rotazione, trasformazione, ...
cpdf_restore($pdf)
?>
```

Vedere anche cpdf_save().

cpdf_rlineto (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Disegna una linea

void **cpdf_rlineto** (int documento pdf, float x-coor, float y-coor [, int modo]) \linebreak

La funzione **cpdf_rlineto()** disegna una linea dal punto corrente al punto con coordinate relative (`x-coor`, `y-coor`).

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_moveto(), cpdf_rmoveto() e cpdf_curveto().

cpdf_rmoveto (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Setta il punto corrente

void **cpdf_rmoveto** (int documento pdf, float x-coor, float y-coor [, int modo]) \linebreak

La funzione **cpdf_rmoveto()** setta il punto corrente alle coordinate relative `x-coor` e `y-coor`.

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_moveto().

cpdf_rotate (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Effettua una rotazione

void **cpdf_rotate** (int documento pdf, float angolo) \linebreak

La funzione **cpdf_rotate()** effettua una rotazione di `angolo` gradi.

cpdf_rotate_text (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Setta l'angolo di rotazione del testo

void **cpdf_rotate_text** (int documento pdf, float angolo) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cpdf_save (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Salva le impostazioni correnti

void **cpdf_save** (int documento pdf) \linebreak

La funzione **cpdf_save()** salva le impostazioni correnti. Funziona come il comando postscript gsave. Molto utile se si vuole traslare o ruotare un oggetto senza intaccare gli altri oggetti.

Vedere anche cpdf_restore().

cpdf_save_to_file (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Salva il documento pdf in un file

void **cpdf_save_to_file** (int documento pdf, string nome del file) \linebreak

La funzione **cpdf_save_to_file()** salva il documento pdf in un file se è stato creato in memoria.

Questa funzione non è necessaria se il documento pdf è stato aperto specificando un nome di un file come parametro di cpdf_open().

Vedere anche cpdf_output_buffer() e cpdf_open().

cpdf_scale (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Applica il coefficiente di scala

void **cpdf_scale** (int documento pdf, float scala-x, float scala-y) \linebreak

La funzione **cpdf_scale()** applica il coefficiente di scala in entrambe le direzioni.

cpdf_set_action_url (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Imposta un link

void **cpdf_set_action_url** (int documento pdf, float xll, float yll, float xur, float xur, string url [, int modo]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cpdf_set_char_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta la spaziatura fra caratteri

void **cpdf_set_char_spacing** (int documento pdf, float spaziatura) \linebreak

La funzione **cpdf_set_char_spacing()** imposta la spaziatura fra caratteri.

Vedere anche cpdf_set_word_spacing() e cpdf_set_leading().

cpdf_set_creator (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta il campo creatore nel documento pdf

void **cpdf_set_creator** (string creatore) \linebreak

La funzione **cpdf_set_creator()** imposta il creatore del documento pdf.

Vedere anche cpdf_set_subject(), cpdf_set_title() e cpdf_set_keywords().

cpdf_set_current_page (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Imposta la pagina corrente

void **cpdf_set_current_page** (int documento pdf, int numero di pagina) \linebreak

La funzione **cpdf_set_current_page()** imposta la pagina sulla quale vengono eseguite tute le operazioni. Ci si può spostare sulle pagine finchè una pagina non è chiusa con cpdf_finalize_page().

Vedere anche cpdf_finalize_page().

cpdf_set_font (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Seleziona l'aspetto e la dimensione del font corrente

void **cpdf_set_font** (int documento pdf, string nome del font, float dimensione, string codifica) \linebreak

La funzione **cpdf_set_font()** imposta l'aspetto, la dimensione e la codifica del font corrente. Attualmente solo i font standard postscript sono supportati.

L'ultimo parametro *`codifica`* può assumere i seguenti valori: "MacRomanEncoding", "MacExpertEncoding", "WinAnsiEncoding", e "`NULL`". "`NULL`" rappresenta la codifica incorporata del font.

Vedere il manuale di ClibPDF per maggiori informazioni, specialmente su come supportare font asiatici.

cpdf_set_font_directories (PHP 4 >= 4.0.6)

Imposta le directory in cui cercare quando si utilizzano font esterni

void **cpdf_set_font_directories** (int documento pdf, string pfmdir, string pfbdir) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cpdf_set_font_map_file (PHP 4 >= 4.0.6)

Imposta la tabella di associazione tra nome di file e nome di font, quando si usano font esterni

void **cpdf_set_font_map_file** (int documento pdf, string nome del file) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cpdf_set_horiz_scaling (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta il fattore di scala orizzontale del testo

void **cpdf_set_horiz_scaling** (int documento pdf, float scala) \linebreak

La funzione **cpdf_set_horiz_scaling()** imposta il fattore di scala orizzontale a *scala* percento.

cpdf_set_keywords (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta i campi chiave del documento pdf

void **cpdf_set_keywords** (string parole chiave) \linebreak

La funzione **cpdf_set_keywords()** imposta le parole chiave di un documento pdf.

Vedere anche cpdf_set_title(), cpdf_set_creator() e cpdf_set_subject().

cpdf_set_leading (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta la distanza fra righe di testo

void **cpdf_set leading** (int documento pdf, float distanza) \linebreak

La funzione **cpdf_set_leading()** imposta la distanza fra righe di testo. Verrà utilizzata se il testo viene scritto con cpdf_continue_text().

Vedere cpdf_continue_text().

cpdf_set_page_animation (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Sets duration between pages

void **cpdf_set_page_animation** (int pdf document, int transition, float duration) \linebreak

The **cpdf_set_page_animation()** function set the transition between following pages.

The value of *transition* can be

0 for none,
1 for two lines sweeping across the screen reveal the page,
2 for multiple lines sweeping across the screen reveal the page,
3 for a box reveals the page,
4 for a single line sweeping across the screen reveals the page,
5 for the old page dissolves to reveal the page,
6 for the dissolve effect moves from one screen edge to another,
7 for the old page is simply replaced by the new page (default)

The value of *duration* is the number of seconds between page flipping.

cpdf_set_subject (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta il campo soggetto del documento pdf

void **cpdf_set_subject** (string soggetto) \linebreak

La funzione **cpdf_set_subject()** imposta il soggetto del documento pdf.

Vedere anche cpdf_set_title(), cpdf_set_creator() e cpdf_set_keywords().

cpdf_set_text_matrix (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta la matrice di testo

void **cpdf_set_text_matrix** (int documento pdf, array matrice) \linebreak

La funzione **cpdf_set_text_matrix()** imposta una matrice che descrive una trasformazione applicata sul font corrente.

cpdf_set_text_pos (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta la posizione del testo

void **cpdf_set_text_pos** (int documento pdf, float x-coor, float y-coor [, int modo]) \linebreak

La funzione **cpdf_set_text_pos()** imposta la posizione del testo per la prossima chiamata della funzione cpdf_show().

Il parametro facoltativo `modo` determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente.

Vedere anche cpdf_show() e cpdf_text().

cpdf_set_text_rendering (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Determines how text is rendered

void **cpdf_set_text_rendering** (int pdf document, int mode) \linebreak

The **cpdf_set_text_rendering()** function determines how text is rendered.

The possible values for `mode` are 0=fill text, 1=stroke text, 2=fill and stroke text, 3=invisible, 4=fill text and add it to clipping path, 5=stroke text and add it to clipping path, 6=fill and stroke text and add it to clipping path, 7=add it to clipping path.

cpdf_set_text_rise (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets the text rise

void **cpdf_set_text_rise** (int pdf document, float value) \linebreak

The **cpdf_set_text_rise()** function sets the text rising to *value* units.

cpdf_set_title (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta il campo titolo del documento pdf

void **cpdf_set_title** (string titolo) \linebreak

La funzione **cpdf_set_title()** imposta il titolo di un documento pdf.

Vedere anche cpdf_set_subject(), cpdf_set_creator() e cpdf_set_keywords().

cpdf_set_viewer_preferences (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

How to show the document in the viewer

void **cpdf_set_viewer_preferences** (int pdfdoc, array preferences) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cpdf_set_word_spacing (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Imposta la spaziatura fra parole

void **cpdf_set_word_spacing** (int documento pdf, float spaziatura) \linebreak

La funzione **cpdf_set_word_spacing()** imposta la spaziatura fra parole.

Vedere cpdf_set_char_spacing() e cpdf_set_leading().

cpdf_setdash (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets dash pattern

void **cpdf_setdash** (int pdf document, float white, float black) \linebreak

The **cpdf_setdash()** function set the dash pattern *white* white units and *black* black units. If both are 0 a solid line is set.

cpdf_setflat (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets flatness

void **cpdf_setflat** (int pdf document, float value) \linebreak

The **cpdf_setflat()** function set the flatness to a value between 0 and 100.

cpdf_setgray (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets drawing and filling color to gray value

void **cpdf_setgray** (int pdf document, float gray value) \linebreak

The **cpdf_setgray()** function sets the current drawing and filling color to the given gray value.

See also cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_setgray_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets filling color to gray value

void **cpdf_setgray_fill** (int pdf document, float value) \linebreak

The **cpdf_setgray_fill()** function sets the current gray value to fill a path.

See also cpdf_setrgbcolor_fill().

cpdf_setgray_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets drawing color to gray value

void **cpdf_setgray_stroke** (int pdf document, float gray value) \linebreak

The **cpdf_setgray_stroke()** function sets the current drawing color to the given gray value.

See also cpdf_setrgbcolor_stroke().

cpdf_setlinecap (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets linecap parameter

void **cpdf_setlinecap** (int pdf document, int value) \linebreak

The **cpdf_setlinecap()** function set the linecap parameter between a value of 0 and 2. 0 = butt end, 1 = round, 2 = projecting square.

cpdf_setlinejoin (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets linejoin parameter

void **cpdf_setlinejoin** (int pdf document, long value) \linebreak

The **cpdf_setlinejoin()** function set the linejoin parameter between a value of 0 and 2. 0 = miter, 1 = round, 2 = bevel.

cpdf_setlinewidth (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets line width

void **cpdf_setlinewidth** (int pdf document, float width) \linebreak

The **cpdf_setlinewidth()** function set the line width to `width`.

cpdf_setmiterlimit (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets miter limit

void **cpdf_setmiterlimit** (int pdf document, float value) \linebreak

The **cpdf_setmiterlimit()** function set the miter limit to a value greater or equal than 1.

cpdf_setrgbcolor (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets drawing and filling color to rgb color value

void **cpdf_setrgbcolor** (int pdf document, float red value, float green value, float blue value) \linebreak

The **cpdf_setrgbcolor()** function sets the current drawing and filling color to the given rgb color value.

See also cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor_fill().

cpdf_setrgbcolor_fill (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets filling color to rgb color value

void **cpdf_setrgbcolor_fill** (int pdf document, float red value, float green value, float blue value) \linebreak

The **cpdf_setrgbcolor_fill()** function sets the current rgb color value to fill a path.

See also cpdf_setrgbcolor_stroke(), cpdf_setrgbcolor().

cpdf_setrgbcolor_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets drawing color to rgb color value

void **cpdf_setrgbcolor_stroke** (int pdf document, float red value, float green value, float blue value) \linebreak

The **cpdf_setrgbcolor_stroke()** function sets the current drawing color to the given rgb color value.

See also cpdf_setrgbcolor_fill(), cpdf_setrgbcolor().

cpdf_show (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Output text at current position

void **cpdf_show** (int pdf document, string text) \linebreak

The **cpdf_show()** function outputs the string in *text* at the current position.

See also cpdf_text(), cpdf_begin_text(), cpdf_end_text().

cpdf_show_xy (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Output text at position

void **cpdf_show_xy** (int pdf document, string text, float x-coor, float y-coor [, int mode]) \linebreak

The **cpdf_show_xy()** function outputs the string *text* at position with coordinates (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

> **Nota:** The function **cpdf_show_xy()** is identical to cpdf_text() without the optional parameters.

See also cpdf_text().

cpdf_stringwidth (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Returns width of text in current font

float **cpdf_stringwidth** (int pdf document, string text) \linebreak

The **cpdf_stringwidth()** function returns the width of the string in *text*. It requires a font to be set before.

See also cpdf_set_font().

cpdf_stroke (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Draw line along path

void **cpdf_stroke** (int pdf document) \linebreak

The **cpdf_stroke()** function draws a line along current path.

See also cpdf_closepath(), cpdf_closepath_stroke().

cpdf_text (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Scrive il testo con i parametri

void **cpdf_text** (int documento pdf, string testo, float x-coor, float y-coor [, int modo [, float orientamento [, int allineamento]]]) \linebreak

La funzione **cpdf_text()** scrive la stringa *testo* alla posizione con coordinate (*x-coor*, *y-coor*).

Il parametro facoltativo *modo* determina l'unità di lunghezza. Se è 0 o è omesso viene utilizzata l'unità di default specificata per la pagina. Altrimenti le coordinate sono misurate in punti postscript trascurando l'unità corrente. Il parametro facoltativo *orientamento* è la rotazione del testo in gradi. Il parametro facoltativo *allineamento* determina come viene allineato il testo.

Vedere la documentazione ClibPDF per i possibili valori.

Vedere anche cpdf_show_xy().

cpdf_translate (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Sets origin of coordinate system

void **cpdf_translate** (int pdf document, float x-coor, float y-coor [, int mode]) \linebreak

The **cpdf_translate()** function set the origin of coordinate system to the point (*x-coor*, *y-coor*).

The optional parameter *mode* determines the unit length. If it's 0 or omitted the default unit as specified for the page is used. Otherwise the coordinates are measured in postscript points disregarding the current unit.

XI. Funzioni di Crack

Queste funzioni permettono di usare la libreria CrackLib per testare la ’forza’ di una password. La ’forza’ di una password è testata attraverso un controllo sulla lunghezza, sull’uso di maiuscole e minuscole ed un controllo attraverso lo specifico dizionario di CrackLib. CrackLib darà anche utili messaggi diagnostici che aiuteranno nel ’rafforzare’ la password.

Requisiti

Maggiori informazioni riguardo CrackLib possono essere trovate, insieme alla libreria, a http://www.users.dircon.co.uk/~crypto/.

Installazione

Per utilizzare queste funzioni, bisogna compilare il PHP con il supporto Crack usando l’opzione `--with-crack[=DIR]`.

Configurazione a Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Tipi di risorsa

Questa estensione non definisce alcuna direttiva di configurazione

Costanti predefinite

Questa estensione non definisce alcuna direttiva di configurazione

Esempio

Questo esempio mostra come aprire un dizionario di CrackLib, testare una determinata password, recuperare ogni messaggio diagnostico e chiudere il dizionario.

Esempio 1. Esempio di CrackLib

```
<?php
// Apre il dizionario di CrackLib
$dizionario = crack_opendict(’/usr/local/lib/pw_dict’)
     or die(’Incapace di aprire il dizionario di CrackLib’);

// Esegue il controllo della password
```

```
$controllo = controllo_crack($dizionario, 'gx9A2s0x');

// Recupera i messaggi
$messaggio = crack_getlastmessage();
echo $messaggio; // 'password forte'

// Chiude il dizionario
crack_closedict($dizionario);
?>
```

Nota: Se crack_check() restituisce TRUE, crack_getlastmessage() restituirà 'password forte'.

crack_check (PHP 4 >= 4.0.5)

Effettua un controllo nascosto con la password data

bool **crack_check** ([resource dizionario, string password]) \linebreak

Restituisce `TRUE` se *`password`* è forte, altrimenti `FALSE`.

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

crack_check() effettua un controllo nascosto della *`password`* data nel *`dizionario`* specificato. Se *`dizionario`* non è specificato viene utilizzato l'ultimo dizionario aperto.

crack_closedict (PHP 4 >= 4.0.5)

Chiude un dizionario di CrackLib aperto

bool **crack_closedict** ([resource dizionario]) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

crack_closedict() chiude l'identificatore del *`dizionario`* specificato. Se *`dizionario`* non è specificato, verrà chiuso il dizionario corrente.

crack_getlastmessage (PHP 4 >= 4.0.5)

Restituisce il messaggio dell'ultimo controllo nascosto

string **crack_getlastmessage** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

crack_getlastmessage() restituisce il messaggio dell'ultimo controllo nascosto.

crack_opendict (PHP 4 >= 4.0.5)

Apre un nuovo dizionario di CrackLib

resource **crack_opendict** (string dizionario) \linebreak

Restituisce un identificatore di risorsa dizionario in caso di successo, o `FALSE` in caso di fallimento.

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

crack_opendict() apre il *`dizionario`* specificato di CrackLib per usarlo con crack_check().

> **Nota:** Può essere aperto solo un dizionario alla volta.

Vedere anche: crack_check() e crack_closedict().

XII. Funzioni CURL, Client URL Library

Introduzione

PHP supporta libcurl, una libreria creata da Daniel Stenberg, che permette di collegarsi e comunicare con parecchi tipi di server e con parecchi tipi di protocolli. Libcurl al momento supporta i protocolli http, https, ftp, gopher, telnet, dict, file, e ldap. libcurl supporta anche i certificati HTTPS, HTTP POST, HTTP PUT, l'upload via FTP (questo può essere ottenuto anche con l'estensione ftp di PHP), upload attraverso una form HTTP, proxy, cookie e autenticazione con utente e password.

Queste funzioni sono state aggiunte nel PHP 4.0.2.

Requisiti

Per utilizzare le funzioni CURL occorre installare il pacchetto CURL (http://curl.haxx.se/). PHP richiede che si usi CURL 7.0.2-beta o successivi. PHP non funzionerà con alcuna versione di CURL antecedente alla 7.0.2-beta.

Istallazione

Al fine di utilizzare il supporto CURL occorre anche compilare PHP con `--with-curl[=DIR]` dove DIR è il percorso della directory che contiene le directory lib e include. Nella directory "include" ci dovrebbe essere una cartella chiamata "curl" che dovrebbe contenere i file easy.h e curl.h. Ci dovrebbe essere un file chiamato "libcurl.a" nella directory "lib".

Costanti Predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata

compilata nel PHP o se è stata caricata dinamicamente a runtime.

`CURLOPT_PORT` (integer)

`CURLOPT_FILE` (integer)

`CURLOPT_INFILE` (integer)

`CURLOPT_INFILESIZE` (integer)

`CURLOPT_URL` (integer)

`CURLOPT_PROXY` (integer)

`CURLOPT_VERBOSE` (integer)

`CURLOPT_HEADER` (integer)

`CURLOPT_HTTPHEADER` (integer)

`CURLOPT_NOPROGRESS` (integer)

`CURLOPT_NOBODY` (integer)

`CURLOPT_FAILONERROR` (integer)

`CURLOPT_UPLOAD` (integer)

`CURLOPT_POST` (integer)

`CURLOPT_FTPLISTONLY` (integer)

`CURLOPT_FTPAPPEND` (integer)

`CURLOPT_NETRC` (integer)

`CURLOPT_FOLLOWLOCATION` (integer)

`CURLOPT_FTPASCII` (integer)

fondo che sta dietro le funzioni CURL è: si inizializza una sessione CURL usando curl_init(), si impostano le opzioni per il trasferimento tramite curl_exec() e quindi si termina la sessione usando curl_close(). Qui di seguito si trova un esempio che fa uso delle funzioni CURL per scaricare la homepage del sito example.com e metterla in un file:

Esempio 1. Usare il modulo CURL di PHP per scaricare la homepage di example.com

```
<?php

$ch = curl_init ("http://www.example.com/");
$fp = fopen ("homepage_example.txt", "w");

curl_setopt ($ch, CURLOPT_FILE, $fp);
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);
curl_close ($ch);
fclose ($fp);
?>
```

curl_close (PHP 4 >= 4.0.2)

Chiude una sessione CURL

void **curl_close** (int ch) \linebreak

Questa funzione chiude una sessione CURL e libera tutte le risorse. L'handle CURL *ch* viene anch'esso eliminato.

curl_errno (PHP 4 >= 4.0.3)

Restituisce un intero contenente il numero dell'ultimo errore

int **curl_errno** (int ch) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

curl_error (PHP 4 >= 4.0.3)

Restituisce una stringa contenente l'ultimo errore relativo alla sessione corrente

string **curl_error** (int ch) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

curl_exec (PHP 4 >= 4.0.2)

Esegue una sessione CURL

bool **curl_exec** (int ch) \linebreak

Questa funzione deve essere chiamata dopo aver inizializzato una sessione CURL e dopo che tutte le opzioni per la sessione sono state impostate. La sua funzione è semplicemente quella di eseguire la sessione CURL predefinita (identificata dal parametro *ch*).

Suggerimento: Come con qualsiasi cosa che invia il risultato direttamente al browser, è possibile utilizzare la funzione output-control per catturare l'uscita di questa funzione e salvarla - per esempio - in una stringa.

curl_getinfo (PHP 4 >= 4.0.4)

Ottiene informazioni relative a un determinato trasferimento

string **curl_getinfo** (int ch, int opt) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

curl_init (PHP 4 >= 4.0.2)

Inizializza una sessione CURL

int **curl_init** ([string url]) \linebreak

curl_init() inizializza una nuova sessione e restituisce un handle CURL da usarsi con le funzioni curl_setopt(), curl_exec() e curl_close(). Se viene dato il parametro opzionale `url`, allora l'opzione CURLOPT_URL verrà impostata al valore di quel parametro. Questo si può impostare manualmente usando la funzione curl_setopt().

Esempio 1. Inizializzare una nuova sessione CURL e scaricare una pagina web

```
<?php
$ch = curl_init();

curl_setopt ($ch, CURLOPT_URL, "http://www.example.com/");
curl_setopt ($ch, CURLOPT_HEADER, 0);

curl_exec ($ch);

curl_close ($ch);
?>
```

Vedere anche: curl_close() e curl_setopt().

curl_setopt (PHP 4 >= 4.0.2)

Imposta una opzione per un trasferimento CURL

bool **curl_setopt** (int ch, string opzione, mixed valore) \linebreak

La funzione **curl_setopt()** imposterà le opzioni per una sessione CURL identificata dal parametro *ch*. il parametro *opzione* è l'opzione che si vuole impostare e *valore* è il valore dell'opzione data da *opzione*.

Il *valore* dovrebbe essere di tipo long per le seguenti opzioni (specificato nel parametro *opzione*):

- *CURLOPT_INFILESIZE*: Quando si carica un file su un server remoto, questa opzione dovrebbe essere usata per dire a PHP quale dimensione attendersi dal file in arrivo.
- *CURLOPT_VERBOSE*: Impostare questa opzione a un valore diverso da zero se si vuole che CURL to segnali tutto quello che sta avvenendo.
- *CURLOPT_HEADER*: Impostare questa opzione a un valore diverso da zero se si vuole che gli header vengano inclusi nell'output.
- *CURLOPT_NOPROGRESS*: Impostare questa opzione a un valore diverso da zero se non si vuole che PHP mostri l'indicazione di avanzamento nei trasferimenti CURL.

  Nota: PHP imposta automaticamente questa opzione a un valore non-zero, esso dovrebbe essere cambiato solo per eseguire il debug.

- *CURLOPT_NOBODY*: Impostare questa opzione a un valore diverso da zero se non si desidera l'inclusione del body nell'output.
- *CURLOPT_FAILONERROR*: Impostare questa opzione a un valore diverso da zero se si vuole che PHP termini senza generare messaggi se il codice HTTP restituito è maggiore di 300. Il comportamento di default è di restituire la pagina comunque, ignorando la presenza del codice.
- *CURLOPT_UPLOAD*: Impostare questa opzione a un valore diverso da zero se si vuole che PHP si prepari per un upload.
- *CURLOPT_POST*: Impostare questa opzione a un valore diverso da zero se si vuole che PHP esegua un comune HTTP POST. Questo POST è di tipo `application/x-www-form-urlencoded`, molto spesso usato nei form HTML.
- *CURLOPT_FTPLISTONLY*: Impostare questa opzione a un valore diverso da zero e PHP semplicemente elencherà i nomi in una directory FTP.
- *CURLOPT_FTPAPPEND*: Impostare questa opzione a un valore diverso da zero e PHP accoderà l'output al file remoto invece di sovrascrivere.
- *CURLOPT_NETRC*: Impostare questa opzione a un valore diverso da zero e PHP leggerà il vostro file `~./netrc` al fine di trovare username e password per il server remoto verso il quale si sta eseguendo la connessione.
- *CURLOPT_FOLLOWLOCATION*: Impostare questa opzione a un valore diverso da zero per seguire ogni header "Location: " che il server dovesse inviare come parte degli header HTTP (si noti che questo è ricorsivo, PHP seguirà tutti gli header "Location: " che dovessero essere inviati.)

- `CURLOPT_PUT`: Impostare questa opzione a un valore diverso da zero per eseguire un HTTP PUT su un file. Il file su cui eseguire PUT deve essere impostato con CURLOPT_INFILE e CURLOPT_INFILESIZE.
- `CURLOPT_MUTE`: Impostare questa opzione a un valore diverso da zero e PHP resterà silenzioso riguardo alle funzioni CURL.
- `CURLOPT_TIMEOUT`: Passa un tipo di dato long come parametro che contiene il tempo massimo, in secondi, che si è disposti a concedere alle funzioni CURL.
- `CURLOPT_LOW_SPEED_LIMIT`: Passa un tipo di dato long come parametro contenente la velocità di trasferimento espressa in byte al secondo, velocità al di sotto della quale il trasferimento dovrà rimanere per CURLOPT_LOW_SPEED_TIME secondi affinché PHP lo consideri troppo lento e lo termini.
- `CURLOPT_LOW_SPEED_TIME`: Passa un tipo di dato long come parametro che contiene il tempo in secondi che il trasferimento dovrà rimanere al di sotto del valore CURLOPT_LOW_SPEED_LIMIT affinché PHP lo consideri troppo lento e lo termini.
- `CURLOPT_RESUME_FROM`: Passa un tipo di dato long come parametro che contiene l'offset, in byte, dal quale partire per effettuare il trasferimento.
- `CURLOPT_SSLVERSION`: Passa un tipo di dato long come parametro che contiene la versione SSL (2 o 3) da usare. Di default PHP cercherà di determinare questo dato sè, ciononostante, in alcuni casi andrà impostato manualmente.
- `CURLOPT_SSL_VERIFYHOST`: Passa un tipo di dato long se CURL deve verificare il common name del peer certificate durante l'handshake SSL. Un valore di 1 denota che si deve cercare l'esistenza del common name, un valore di 2 denota che dovremo assicurarci che corrisponda all'hostname fornito.
- `CURLOPT_TIMECONDITION`: Passa un tipo di dato long come parametro che che definisce come trattare CURLOPT_TIMEVALUE. Si può impostare questo parametro a TIMECOND_IFMODSINCE o TIMECOND_ISUNMODSINCE. Questa è una caratteristica del solo HTTP.
- `CURLOPT_TIMEVALUE`: Passa un tipo di dato long come parametro che rappresenta il tempo espresso in secondi trascorsi dal 1 Gennaio 1970. Il tempo verrà usato come specificato dall'opzione CURLOPT_TIMEVALUE o di default verrà usato TIMECOND_IFMODSINCE.
- `CURLOPT_RETURNTRANSFER`: Passa un valore diverso da zero se si desidera che CURL restituisca direttamente il trasferimento invece di stamparlo.

il parametro `valole` deve essere una stringa per i seguenti valori del parametro `opzione`:

- `CURLOPT_URL`: Questo è l'URL che si desidera far salvare da PHP. Si può anche impostare questa opzione quando si inizializza una sessione con la funzione curl_init().
- `CURLOPT_USERPWD`: Passa una stringa formattata come [username]:[password], che sarà usata da PHP per la connessione.
- `CURLOPT_PROXYUSERPWD`: Passa una stringa formattata come [username]:[password], per la connessione al proxy HTTP.
- `CURLOPT_RANGE`: Passa il range richiesto. Deve essere nel formato "X-Y", dove X o Y possono essere omessi. I trasferimenti HTTP supportano anche intervalli multipli, separati da virgole come in X-Y,N-M.

- *CURLOPT_POSTFIELDS*: Passa una stringa contenente tutti i dati da inviare durante una richiesta HTTP "POST".
- *CURLOPT_REFERER*: Passa una stringa contenente l'header "referer" che verrà usato in una richiesta HTTP.
- *CURLOPT_USERAGENT*: Passa una stringa contenente l'header "user-agent" che sarà usato nella richiesta HTTP.
- *CURLOPT_FTPPORT*: Passa una stringa contenente il valore che verrà usato per ottenere l'indirizzo IP da usarsi per l'istruzione "POST" dell'ftp. L'istruzione POST dice al server remoto di collegarsi al nostro indirizzo IP specificato. La stringa può essere un semplice indirizzo IP, un hostname, il nome di una interfaccia di rete (sotto UNIX) o semplicemente un '-' per indicare di usare l'indirizzo IP di default del sistema.
- *CURLOPT_COOKIE*: Passa una stringa contenente il contenuto del cookie da impostare nell'header HTTP.
- *CURLOPT_SSLCERT*: Passa una stringa contenente il filename del certificato formattato secondo PEM.
- *CURLOPT_SSLCERTPASSWD*: Passa una stringa contenente la password richiesta per usare il certificato CURLOPT_SSLCERT.
- *CURLOPT_COOKIEFILE*: Passa una stringa contenente il nome del file contenente i dati relativi al cookie. Il file del cookie può essere in formato Netscape o semplicemente un dump degli header in stile HTTP.
- *CURLOPT_CUSTOMREQUEST*: Passa una stringa da usare al posto di `GET` o `HEAD` nell'effettuare una richiesta HTTP. Questo è utile per effettuare un `DELETE` o altre, più oscure richieste HTTP. Valori validi sono cose del tipo `GET`, `POST` e così via; per esempio non inserire qui una richiesta HTTP intera. Per esempio, inserire 'GET /index.html HTTP/1.0\r\n\r\n' sarebbe sbagliato.

  Nota: Non eseguire se non si è sicuri che il proprio server supporti il comando.

- *CURLOPT_PROXY*: Indica il nome del server proxy HTTP attraverso il quale inviare le richieste.
- *CURLOPT_INTERFACE*: Passa il nome dell'interfaccia di rete da usare in uscita. Questo può essere il nome di un'interfaccia, un indirizzo IP o un nome di host.
- *CURLOPT_KRB4LEVEL*: Passa il security level KRB4 (Kerberos 4). Ognuna delle seguenti stringhe (in ordine dalla meno alla più potente): 'clear', 'safe', 'confidential', 'private'. Se la stringa inviata non corrisponde a nessuna di queste, verrà usata 'private'. Se si imposta a NULL, la security KRB4 verrà disabilitata. La security KRB4, al momento, funziona solamente con le transazioni FTP.
- *CURLOPT_HTTPHEADER*: Passa un array di campi header HTTP da impostare.
- *CURLOPT_QUOTE*: Passa un array di comandi FTP da eseguire sul server, prima della richiesta FTP.
- *CURLOPT_POSTQUOTE*: Passa un array di comandi FTP da eseguire sul server, dopo che la richiesta FTP è stata eseguita.

Le opzioni seguenti richiedono un descrittore a file che è ottenuto usando la funzione fopen():

- *CURLOPT_FILE*: Il file dove mettere l'output del trasferimento, di default è STDOUT.

- *CURLOPT_INFILE*: Il file da cui proviene l'input del trasferimento.
- *CURLOPT_WRITEHEADER*: Il file sul quale scrivere la parte di output contenente gli header.
- *CURLOPT_STDERR*: Il file sul quale scrivere gli errori invece di stderr.

curl_version (PHP 4 >= 4.0.2)

Restituisce la versione di CURL in uso

string **curl_version** (void) \linebreak

La funzione **curl_version()** restituisce una stringa contenente la versione di CURL attualmente in uso.

XIII. Funzioni di pagamento Cybercash

Istallazione

Queste funzione sono disponibili solo se l'interprete è stato compilato con l'opzione `--with-cybercash=[DIR]`.

cybercash_base64_decode (PHP 4 >= 4.0.0)

Decodifica dei dati in base64 per Cybercash

string **cybercash_base64_decode** (string inbuff) \linebreak

cybercash_base64_encode (PHP 4 >= 4.0.0)

Codifica dei dati in base64 per Cybercash

string **cybercash_base64_encode** (string inbuff) \linebreak

cybercash_decr (PHP 4 >= 4.0.0)

Decrifrazione Cybercash

array **cybercash_decr** (string wmk, string sk, string inbuff) \linebreak

La funzione restituisce un array associativo con gli elementi "errcode" e, se "errcode" è `FALSE`, "outbuff" (stringa), "outLth" (long) and "macbuff" (stringa).

cybercash_encr (PHP 4 >= 4.0.0)

Criptazione Cybercash

array **cybercash_encr** (string wmk, string sk, string inbuff) \linebreak

La funzione restituisce un array associativo con gli elementi "errcode" e, se "errcode" è `FALSE`, "outbuff" (stringa), "outLth" (long) and "macbuff" (stringa).

XIV. Crédit Mutuel CyberMUT functions

Questa estensione vi permette di processare le transazioni delle carte di credito usando il sistema Crédit Mutuel CyberMUT (http://www.creditmutuel.fr/centre_commercial/vendez_sur_internet.html).

CynerMUT, in Francia, è un popolare servizio di pagamento tramite web, fornito dalla banca Crédit Mutuel. Se siete stranieri in Francia, queste funzioni non saranno utili.

Queste funzioni sono disponibili se PHP è stato compilato con l'opzione `--with-cybermut[=DIR]`, dove DIR è la locazione di `libcm-mac.a` e `cm-mac.h`. Verrà richiesto l'appropriato SDK per la piattaforma, che sarà spedito dopo la sottoscrizione al servizio CyberMUT (contattateli via Web, o recatevi presso il più vicino Crédit Mutuel).

L'uso di queste funzioni è abbastanza simile alle funzioni originali, ad eccezione dei parametri restituiti da cybermut_creerformulairecm() e cybermut_creerreponsecm(), che sono restituiti direttamente dalle funzioni PHP, mentre loro sono passati in riferimento nelle funzioni originali.

Queste funzioni sono state aggiunte in PHP 4.0.6.

> **Nota:** Queste funzioni forniscono solo un link all'SDK di CyberMUT. Assicuratevi di leggere la Guida per gli Sviluppatori del CyberMUT per i dettagli completi dei parametri richiesti.

cybermut_creerformulairecm (PHP 4 >= 4.0.5)

Genera un form HTML per la richiesta di pagamento

string **cybermut_creerformulairecm** (string url_CM, string version, string TPE, string montant, string ref_commande, string texte_libre, string url_retour, string url_retour_ok, string url_retour_err, string langue, string code_societe, string texte_bouton) \linebreak

cybermut_creerformulairecm() è usato per generare il form HTML per la richiesta di pagamento.

Esempio 1. Primo passo di pagamento (equiv cgi1.c)

```
<?php
// Directory dove si trovano le chiavi
putenv("CMKEYDIR=/var/creditmut/cles");

// Numero Versione
$VERSION="1.2";

  $retour =  cybermut_creerformulairecm(
  "https://www.creditmutuel.fr/test/telepaiement/paiement.cgi",
  $VERSION,
  "1234567890",
  "300FRF",
  $REFERENCE,
  $TEXTE_LIBRE,
  $URL_RETOUR,
  $URL_RETOUR_OK,
  $URL_RETOUR_ERR,
  "francais",
  "company",
  "Paiement par carte bancaire");

  echo $retour;
?>
```

Vedere anche cybermut_testmac() e cybermut_creerreponsecm().

cybermut_creerreponsecm (PHP 4 >= 4.0.5)

Genera la conferma della consegna della conferma di pagamento

string **cybermut_creerreponsecm** (string phrase) \linebreak

cybermut_creerreponsecm() restituisce una stringa contenente un messaggio di conferma di consegna.

Il parametro è "OK" se il messaggio di conferma del pagamento è stato correttamente autenticato da cybermut_testmac(). Ogni altra catena è considerata come un messaggio di errore.

Vedere anche cybermut_creerformulairecm() e cybermut_testmac().

cybermut_testmac (PHP 4 >= 4.0.5)

Assicura che non siano contenuti dati manipolati nel messaggio di conferma ricevuto

bool **cybermut_testmac** (string code_MAC, string version, string TPE, string cdate, string montant, string ref_commande, string texte_libre, string code-retour) \linebreak

cybermut_testmac() è usato per assicurare che non siano contenuti dati manipolati nel messaggio di conferma ricevuto. Prestate attenzione ai parametri *code-retour* e *texte-libre*, che non possono essere valutati tal quali, a causa del trattino. Dovete recuperarli usando:

```
<?php
  $code_retour=$HTTP_GET_VARS["code-retour"];
  $texte_libre=$HTTP_GET_VARS["texte-libre"];
?>
```

Esempio 1. Ultimo passaggio del pagamento (equiv cgi2.c)

```
<?php
// Make sure that Enable Track Vars is ON.
// Directory where are located the keys
putenv("CMKEYDIR=/var/creditmut/cles");

// Version number
$VERSION="1.2";

$texte_libre = $HTTP_GET_VARS["texte-libre"];
$code_retour = $HTTP_GET_VARS["code-retour"];

$mac_ok = cybermut_testmac($MAC,$VERSION,$TPE,$date,$montant,$reference,$texte_libre,$coc

if ($mac_ok) {

  //
  // insert data processing here
  //
  //

  $result=cybermut_creerreponsecm("OK");
} else {
  $result=cybermut_creerreponsecm("Document Falsifie");
}

?>
```

Vedere anche cybermut_creerformulairecm() e cybermut_creerreponsecm().

XV. Cyrus IMAP administration functions

Introduzione

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

Costanti Predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata compilata nel PHP o se è stata caricata dinamicamente a runtime.

`CYRUS_CONN_NONSYNCLITERAL` (integer)

`CYRUS_CONN_INITIALRESPONSE` (integer)

`CYRUS_CALLBACK_NUMBERED` (integer)

`CYRUS_CALLBACK_NOLITERAL` (integer)

cyrus_authenticate (PHP 4 >= 4.1.0)

Authenticate agaings a Cyrus IMAP server

bool **cyrus_authenticate** (resource connection [, string mechlist [, string service [, string user [, int minssf [, int maxssf]]]]]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cyrus_bind (PHP 4 >= 4.1.0)

Bind callbacks to a Cyrus IMAP connection

bool **cyrus_bind** (resource connection, array callbacks) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cyrus_close (PHP 4 >= 4.1.0)

Close connection to a cyrus server

bool **cyrus_close** (resource connection) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cyrus_connect (PHP 4 >= 4.1.0)

Connect to a Cyrus IMAP server

resource **cyrus_connect** ([string host [, string port [, int flags]]]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cyrus_query (PHP 4 >= 4.1.0)

Send a query to a Cyrus IMAP server

bool **cyrus_query** (resource connection, string query) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

cyrus_unbind (PHP 4 >= 4.1.0)

Unbind ...

bool **cyrus_unbind** (resource connection, string trigger_name) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

XVI. Funzioni di tipo dei caratteri

Le funzioni fornite da questa estensione controllano se un carattere o una stringa rientrano in una classe di caratteri in accordo con l'ambiente corrente (vedere anche setlocale()).

Quando vengono chiamate con un numero intero come argomento queste funzioni si comportano esattamente come il loro equivalente in C presente in "ctype.h".

Quando vengono chiamate con una stringa come argomento controlleranno ogni carattere della stringa e ritorneranno `TRUE` se ogni carattere della stringa soddisfa il criterio richiesto.

Passare qualsiasi cosa eccetto una stringa o un numero intero restituirà immediatamenente `FALSE`.

Requisiti

Nessuno oltre alle funzioni della libreria standard del C che sono sempre disponibili.

Installazione

A partire da PHP 4.2.0 queste funzioni sono abilitate di default. Per le versioni più vecchie bisogna configurare e compilare PHP con `--enable-ctype`.

Configurazione a Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Tipi di risorsa

Questa estensione non definisce alcuna direttiva di configurazione

Costanti predefinite

Questa estensione non definisce alcuna direttiva di configurazione

ctype_alnum (PHP 4 >= 4.0.4)

Controlla i caratteri alfanumerici

bool **ctype_alnum** (string testo) \linebreak

Restituisce `TRUE` se ogni carattere di *`testo`* è una lettera o una cifra, `FALSE` in caso contrario. Nell'ambiente standard del `C` le lettere sono solamente `[A-Za-z]`. La funzione è equivalente a `(ctype_alpha($text) || ctype_digit($text))`.

Vedere anche ctype_alpha(), ctype_digit() e setlocale().

ctype_alpha (PHP 4 >= 4.0.4)

Controlla i caratteri alfabetici

bool **ctype_alpha** (string testo) \linebreak

Restituisce `TRUE` se ogni carattere di *`testo`* è una lettera dell'ambiente corrente, `FALSE` in caso contrario. Nell'ambiente standard del `C` le lettere sono solo `[A-Za-z]` e **ctype_alpha()** è equivalente a `(ctype_upper($text) || ctype_lower($text))`, ma altri linguaggi hanno lettere che non sono considerate nè maiuscole nè minuscole.

Vedere anche ctype_upper(), ctype_lower() e setlocale().

ctype_cntrl (PHP 4 >= 4.0.4)

Controlla i caratteri di controllo

bool **ctype_cntrl** (string testo) \linebreak

Restituisce `TRUE` se ogni carattere di *`testo`* ha una speciale funzione di controllo, `FALSE` in caso contrario. I caratteri di controllo sono per esempio line feed (avanza di una riga), tab, esc.

ctype_digit (PHP 4 >= 4.0.4)

Controlla i caratteri numerici

bool **ctype_digit** (string testo) \linebreak

Restituisce `TRUE` se ogni carattere di *`testo`* è una cifra decimale, `FALSE` in caso contrario.

Vedere anche ctype_alnum() e ctype_xdigit().

ctype_graph (PHP 4 >= 4.0.4)

Controlla ogni carattere stampabile tranne lo spazio

bool **ctype_graph** (string testo) \linebreak

Restituisce TRUE se ogni carattere di *testo* è stampabile e crea un output realmente visibile (senza spazi bianchi), FALSE in caso contrario.

Vedere anche ctype_alnum(), ctype_print() e ctype_punct().

ctype_lower (PHP 4 >= 4.0.4)

Controlla i caratteri minuscoli

bool **ctype_lower** (string testo) \linebreak

Restituisce TRUE se ogni carattere di *testo* è una lettera minuscola nell'ambiente corrente.

Vedere anche ctype_alpha() e ctype_upper().

ctype_print (PHP 4 >= 4.0.4)

Controlla i caratteri stampabili

bool **ctype_print** (string testo) \linebreak

Restituisce TRUE se ogni carattere di *testo* creerà veramente un output (compresi gli spazi). Restituisce FALSE se *testo* contiene dei caratteri di controllo o caratteri che non hanno nessun output o che non hanno per niente una funzione di controllo.

Vedere anche ctype_cntrl(), ctype_graph() e ctype_punct().

ctype_punct (PHP 4 >= 4.0.4)

Controlla ogni carattere stampabile che non è uno spazio o un carattere alfanumerico

bool **ctype_punct** (string testo) \linebreak

Restituisce TRUE se ogni carattere di *testo* è stampabile, ma non è nè una lettera nè; una cifra nè uno spazio, FALSE in caso contrario.

Vedere anche ctype_cntrl(), ctype_graph() e **ctype_punct()**.

ctype_space (PHP 4 >= 4.0.4)

Controlla gli spazi

bool **ctype_space** (string testo) \linebreak

Restituisce TRUE se ogni carattere di *testo* crea qualche tipo di spazio, FALSE in caso contrario. Oltre allo spazio questo include anche tab, tab verticale, line feed (avanza di una riga), carriage return (a capo) e form feed (avanza di un modulo).

ctype_upper (PHP 4 >= 4.0.4)

Controlla i caratteri maiuscoli

bool **ctype_upper** (string testo) \linebreak

Restituisce `TRUE` se ogni carattere di *`testo`* è una lettera maiuscola nell'ambiente corrente.

Vedere anche ctype_alpha() e ctype_lower().

ctype_xdigit (PHP 4 >= 4.0.4)

Controlla i caratteri che rappresentano una cifra esadecimale

bool **ctype_xdigit** (string testo) \linebreak

Restituisce `TRUE` se ogni carattere di *`testo`* è una 'cifra' esadecimale, cioè una cifra decimale o un carattere fra `[A-Fa-f]` , `FALSE` in caso contrario.

Vedere anche ctype_digit().

XVII. Database (dbm-style) abstraction layer functions

Introduzione

These functions build the foundation for accessing Berkeley DB style databases.

This is a general abstraction layer for several file-based databases. As such, functionality is limited to a common subset of features supported by modern databases such as Sleepycat Software's DB2 (http://www.sleepycat.com/). (This is not to be confused with IBM's DB2 software, which is supported through the ODBC functions.)

Requisiti

The behaviour of various aspects depends on the implementation of the underlying database. Functions such as dba_optimize() and dba_sync() will do what they promise for one database and will do nothing for others. You have to download and install supported dba-Handlers.

Tabella 1. List of DBA handlers

Handler	Notes
`dbm`	Dbm is the oldest (original) type of Berkeley DB style databases. You should avoid it, if possible. We do not support the compatibility functions built into DB2 and gdbm, because they are only compatible on the source code level, but cannot handle the original dbm format.
`ndbm`	Ndbm is a newer type and more flexible than dbm. It still has most of the arbitrary limits of dbm (therefore it is deprecated).
`gdbm`	Gdbm is the GNU database manager (ftp://ftp.gnu.org/pub/gnu/gdbm/).
`db2`	DB2 is Sleepycat Software's DB2 (http://www.sleepycat.com/). It is described as "a programmatic toolkit that provides high-performance built-in database support for both standalone and client/server applications.
`db3`	DB3 is Sleepycat Software's DB3 (http://www.sleepycat.com/).
`cdb`	Cdb is "a fast, reliable, lightweight package for creating and reading constant databases." It is from the author of qmail and can be found here (http://cr.yp.to/cdb.html). Since it is constant, we support only reading operations.

When invoking the dba_open() or dba_popen() functions, one of the handler names must be supplied as an argument. The actually available list of handlers is displayed by invoking phpinfo().

Istallazione

By using the `--enable-dba=shared` configuration option you can build a dynamic loadable modul to enable PHP for basic support of dbm-style databases. You also have to add support for at least one of the following handlers by specifying the `--with-XXXX` configure switch to your PHP configure line.

Tabella 2. Supported DBA handlers

Handler	Configure Switch
`dbm`	To enable support for dbm add `--with-dbm[=DIR]`.
`ndbm`	To enable support for ndbm add `--with-ndbm[=DIR]`.
`gdbm`	To enable support for gdbm add `--with-gdbm[=DIR]`.
`db2`	To enable support for db2 add `--with-db2[=DIR]`.
`db3`	To enable support for db3 add `--with-db3[=DIR]`.
`cdb`	To enable support for cdb add `--with-cdb[=DIR]`.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

The functions dba_open() and dba_popen() return a handle to the specified database file to access

which is used by all other dba-function calls.

Costanti Predefinite

Questa estensione non definisce alcuna costante.

Esempi

Esempio 1. DBA example

```
<?php

$id = dba_open ("/tmp/test.db", "n", "db2");

if (!$id) {
    echo "dba_open failed\n";
    exit;
}

dba_replace ("key", "This is an example!", $id);

if (dba_exists ("key", $id)) {
    echo dba_fetch ("key", $id);
    dba_delete ("key", $id);
}

dba_close ($id);
?>
```

DBA is binary safe and does not have any arbitrary limits. However, it inherits all limits set by the underlying database implementation.

All file-based databases must provide a way of setting the file mode of a new created database, if that is possible at all. The file mode is commonly passed as the fourth argument to dba_open() or dba_popen().

You can access all entries of a database in a linear way by using the dba_firstkey() and dba_nextkey() functions. You may not change the database while traversing it.

Esempio 2. Traversing a database

```
<?php

// ...open database...

$key = dba_firstkey ($id);
```

```
while ($key != false) {
    if (...) {          // remember the key to perform some action later
        $handle_later[] = $key;
    }
    $key = dba_nextkey ($id);
}

for ($i = 0; $i < count($handle_later); $i++)
    dba_delete ($handle_later[$i], $id);

?>
```

dba_close (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Close database

void **dba_close** (resource handle) \linebreak

dba_close() closes the established database and frees all resources specified by *handle*.

handle is a database handle returned by dba_open().

dba_close() does not return any value.

See also: dba_open() and dba_popen()

dba_delete (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Delete entry specified by key

bool **dba_delete** (string key, resource handle) \linebreak

dba_delete() deletes the entry specified by *key* from the database specified with *handle*.

key is the key of the entry which is deleted.

handle is a database handle returned by dba_open().

dba_delete() returns `TRUE` or `FALSE`, if the entry is deleted or not deleted, respectively.

See also: dba_exists(), dba_fetch(), dba_insert(), and dba_replace().

dba_exists (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Check whether key exists

bool **dba_exists** (string key, resource handle) \linebreak

dba_exists() checks whether the specified *key* exists in the database specified by *handle*.

Key is the key the check is performed for.

Handle is a database handle returned by dba_open().

dba_exists() returns `TRUE` or `FALSE`, if the key is found or not found, respectively.

See also: dba_fetch(), dba_delete(), dba_insert(), and dba_replace().

dba_fetch (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fetch data specified by key

string **dba_fetch** (string key, resource handle) \linebreak

dba_fetch() fetches the data specified by *key* from the database specified with *handle*.

Key is the key the data is specified by.

Handle is a database handle returned by dba_open().

dba_fetch() returns the associated string or FALSE, if the key/data pair is found or not found, respectively.

See also: dba_exists(), dba_delete(), dba_insert(), and dba_replace().

dba_firstkey (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fetch first key

string **dba_firstkey** (resource handle) \linebreak

dba_firstkey() returns the first key of the database specified by *handle* and resets the internal key pointer. This permits a linear search through the whole database.

Handle is a database handle returned by dba_open().

dba_firstkey() returns the key or FALSE depending on whether it succeeds or fails, respectively.

See also: dba_nextkey() and example 2 in the DBA examples

dba_insert (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Insert entry

bool **dba_insert** (string key, string value, resource handle) \linebreak

dba_insert() inserts the entry described with *key* and *value* into the database specified by *handle*. It fails, if an entry with the same *key* already exists.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by dba_open().

dba_insert() returns TRUE or FALSE, depending on whether it succeeds of fails, respectively.

See also: dba_exists() dba_delete() dba_fetch() dba_replace()

dba_nextkey (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Fetch next key

string **dba_nextkey** (resource handle) \linebreak

dba_nextkey() returns the next key of the database specified by *handle* and advances the internal key pointer.

handle is a database handle returned by dba_open().

dba_nextkey() returns the key or FALSE depending on whether it succeeds or fails, respectively.

See also: dba_firstkey() and example 2 in the DBA examples

dba_open (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Open database

resource **dba_open** (string path, string mode, string handler [, ...]) \linebreak

dba_open() establishes a database instance for `path` with `mode` using `handler`.

`path` is commonly a regular path in your filesystem.

`mode` is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

`handler` is the name of the handler which shall be used for accessing `path`. It is passed all optional parameters given to **dba_open()** and can act on behalf of them.

dba_open() returns a positive handle or `FALSE`, in the case the open is successful or fails, respectively.

See also: dba_popen() dba_close()

dba_optimize (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Optimize database

bool **dba_optimize** (resource handle) \linebreak

dba_optimize() optimizes the underlying database specified by `handle`.

`handle` is a database handle returned by dba_open().

dba_optimize() returns `TRUE` or `FALSE`, if the optimization succeeds or fails, respectively.

See also: dba_sync()

dba_popen (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Open database persistently

resource **dba_popen** (string path, string mode, string handler [, ...]) \linebreak

dba_popen() establishes a persistent database instance for `path` with `mode` using `handler`.

`path` is commonly a regular path in your filesystem.

`mode` is "r" for read access, "w" for read/write access to an already existing database, "c" for read/write access and database creation if it doesn't currently exist, and "n" for create, truncate and read/write access.

`handler` is the name of the handler which shall be used for accessing `path`. It is passed all optional parameters given to **dba_popen()** and can act on behalf of them.

dba_popen() returns a positive handle or `FALSE`, in the case the open is successful or fails, respectively.

See also: dba_open() dba_close()

dba_replace (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Replace or insert entry

bool **dba_replace** (string key, string value, resource handle) \linebreak

dba_replace() replaces or inserts the entry described with *key* and *value* into the database specified by *handle*.

key is the key of the entry to be inserted.

value is the value to be inserted.

handle is a database handle returned by dba_open().

dba_replace() returns TRUE or FALSE, depending on whether it succeeds of fails, respectively.

See also: dba_exists(), dba_delete(), dba_fetch(), and dba_insert().

dba_sync (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Synchronize database

bool **dba_sync** (resource handle) \linebreak

dba_sync() synchronizes the database specified by *handle*. This will probably trigger a physical write to disk, if supported.

handle is a database handle returned by dba_open().

dba_sync() returns TRUE or FALSE, if the synchronization succeeds or fails, respectively.

See also: dba_optimize()

XVIII. Funzioni di Data e Ora

Introduzione

Puoi usare queste funzione per la data e l'ora. Queste funzioni ti permettono di scaricare la data e l'orario dal server dove gira il PHP. Puoi usare queste funzioni per formattare l'output delle date e degli orari in diversi modi.

> **Nota:** Ricordati che queste funzioni dipendono dai settaggi locali del tuo server. Considera specialmente l'ora legale e gli anni bisestili.

Requisiti

Queste funzioni sono disponibili nei moduli standard, che sono sempre disponibili.

Istallazione

Non ‘ necessaria nessuna installazione per usare queste funzioni, esse fanno parte del core di PHP.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

Questa estensione non definisce alcun tipo di risorsa.

Costanti Predefinite

Questa estensione non definisce alcuna costante.

checkdate (PHP 3, PHP 4 >= 4.0.0)

Verifica una data/orario gragoriana/o

bool **checkdate** (int mese, int giorno, int anno) \linebreak

Restituisce `TRUE` se la data inserita è valida; altrimenti restituisce `FALSE`. Controlla la validità di una data formata dagli argomenti. Una data è considerata valida se:

- anno è compreso tra 1 e 32767
- mese è compreso tra 1 e 12
- *`Giorno`* è compreso tra il numero dei giorni possibile per il *`mese`* dato. Gli *`anno`*(i) bisestili sono presi in considerazione.

Guarda anche mktime() e strtotime().

date (PHP 3, PHP 4 >= 4.0.0)

Formatta una data o orario locale

string **data** (string formato [, int timestamp]) \linebreak

Restituisce una stringa formattata in accordo con il formato della stringa usato nell' intero *`timestamp`* o nell'attuale orario locale se timestamp non è assegnato.

> **Nota:** Il valido intervallo del timestamp è abitualmente da Fri, 13 Dec 1901 20:45:54 GMT a Tue, 19 Jan 2038 03:14:07 GMT. (Queste date corrispondono al valore minimo e al massimo per un intero segnato a 32-bit.)
>
> Per generare un timestamp da una stringa rappresentante la data, devi sapere usare strtotime(). In aggiunta, dei databases hanno funzioni che convertono i loro formati di data in timestamps (come la funzione di MySQL, UNIX_TIMESTAMP).

I seguenti caratteri sono utilizzati nella stringa formato:

- a - "am" o "pm"
- A - "AM" o "PM"
- B - Swatch Internet time
- d - giorno del mese, 2 cifre senza tralasciare gli zero; i.e. "01" a "31"
- D - giorno della settimana, testuale, 3 lettere; i.e. "Fri"
- F - mese, testuale, long; i.e. "January"
- g - ora, formato a 12-ore senza eventuali zero; i.e. "1" a "12"
- G - ora, formato a 24-ore senza eventuali zero; i.e. "0" a "23"
- h - ora, formato a 12-ore; i.e. "01" a "12"
- H - ora, formato a 24-ore; i.e. "00" a "23"

- i - minuti; i.e. "00" a "59"
- I (i grande) - "1" se c'è l'ora legale, "0" altrimenti.
- j - giorno del mese senza eventuali zero; i.e. "1" a "31"
- l ('L' piccola) - giorno della settimana, testuale, long; i.e. "Friday"
- L - valore booleano per stabilire se è un anno bisestile; i.e. "0" o "1"
- m - mese; i.e. "01" a "12"
- M - mese, testuale, 3 lettere; i.e. "Jan"
- n - mese senza eventuali zero; i.e. "1" a "12"
- O - Differenza in ore dal fuso orario Greenwich; i.e. "+0200"
- r - Data formattata RFC 822; i.e. "Thu, 21 Dec 2000 16:01:07 +0200" (aggiunto nel PHP 4.0.4)
- s - secondi; i.e. "00" a "59"
- S - Suffisso ordinale Inglese per i giorni del mese, 2 caratteri; i.e. "th", "nd"
- t - numero di giorni del mese dato; i.e. "28" a "31"
- T - Fuso orario di questo computer; i.e. "MDT"
- U - secondi dall'epoca since the epoch
- w - giorno del mese, numerico, i.e. "0" (Domenica) a "6" (Sabato)
- W - ISO-8601 Numero della settimana dell'anno, le settimane iniziano il lunedì (aggiunto nel PHP 4.1.0) (Sabato)
- Y - anno, 4 cifre; i.e. "1999"
- y - anno, 2 cifre; i.e. "99"
- z - giorno dell'anno; i.e. "0" a "365"
- Z - Fuso orario in secondi (i.e. "-43200" a "43200"). Il fuso orario ad ovest dell'UTC è sempre negativo, e per quelli ad est è sempre positivo.

Caratteri non utilizzati nella stringa del formato saranno scritti come sono. Il formato "Z" restituirà sempre "0" quando si usa gmdate().

Esempio 1. Esempio di data()

```
echo date ("l dS of F Y h:i:s A");
echo "July 1, 2000 is on a " . date ("l", mktime(0,0,0,7,1,2000));
```

Puoi utilizzare un carattere utilizzabile nella stringa del formato come carattere normale facendolo semplicemente precedere dal carattere di escape, il backslash. Se il carattere con un backslash è ancora una sequenza speciale, devi inserire di nuovo il carattere di escape per il backslash.

Esempio 2. Caratteri di escape in date()

```
echo date("l \\t\h\e jS"); // scrive qualcosa tipo 'Saturday the 8th'
```

È possibile usare **date()** e mktime() assieme per cercare delle date nel futuro o nel passato.

Esempio 3. Esempio di date() e mktime()

```
$tomorrow  = mktime (0,0,0,date("m")  ,date("d")+1,date("Y"));
$lastmonth = mktime (0,0,0,date("m")-1,date("d"),  date("Y"));
$nextyear  = mktime (0,0,0,date("m"),  date("d"),  date("Y")+1);
```

Nota: Questo può essere più affidabile della semplice addizione e sottrazione del numero di secondi in un giorno o mese a un timestamp a causa del daylight savings time.

Alcuni esempi della formattazione di **date()**. Nota che puoi scrivere qualsiasi altro carattere senza escape, il quale attualmente non dovrebbe produrre risultati indesiderati, ma altri caratteri potrebbero essere assegnati come caratteri della stringa di formattazione nelle prossime versioni del PHP. Quando si utilizza il carattere di escape, assicurati di usare un singolo apice per prevenire che caratteri come \n facciano iniziare una nuova linea.

Esempio 4. date() Formatting

```
/* Today is March 10th, 2001, 5:16:18 pm */
$today = date("F j, Y, g:i a");                 // March 10, 2001, 5:16 pm
$today = date("m.d.y");                         // 03.10.01
$today = date("j, n, Y");                       // 10, 3, 2001
$today = date("Ymd");                           // 20010310
$today = date('h-i-s, j-m-y, it is w Day z ');  // 05-16-17, 10-03-01, 1631 1618 6 Fripm0
$today = date('\i\t \i\s \t\h\e jS \d\a\y.');   // It is the 10th day.
$today = date("D M j G:i:s T Y");               // Sat Mar 10 15:16:08 MST 2001
$today = date('H:m:s \m \i\s\ \m\o\n\t\h');     // 17:03:17 m is month
$today = date("H:i:s");                         // 17:16:17
```

Per formattare le date in lingue diverse dall'inglese, dovresti usare le funzioni setlocale() e strftime().

Guarda anche getlastmod(), gmdate(), mktime(), strftime() e time().

getdate (PHP 3, PHP 4 >= 4.0.0)

Riceve informazioni su data/orario

array **getdate** ([int timestamp]) \linebreak

Restituisce un array associativo contenente le informazioni sulla data del *timestamp*, o dell'attuale orario locale se non è stato assegnato timestamp, con i seguenti elementi di array:

- "seconds" - secondi
- "minutes" - minuti
- "hours" - ore
- "mday" - giorno del mese
- "wday" - giorno della settimana, numerico : da 0 come Domenica a 6 come Sabato
- "mon" - mese, numerico
- "year" - anno, numerico
- "yday" - giorno dell'anno, numerico; i.e. "299"
- "weekday" - giorno della settimana, testuale, per intero; i.e. "Friday"
- "month" - mese, testuale, per intero; i.e. "January"

Esempio 1. Esempio di getdate()

```
$today = getdate();
$month = $today['month'];
$mday = $today['mday'];
$year = $today['year'];
echo "$month $mday, $year";
```

gettimeofday (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Riceve l'orario attuale

array **gettimeofday** (void) \linebreak

Questa è un'interfaccia di gettimeofday(2). Restituisce un array associativo contenente i dati restituiti dalla chiamata al sistema.

- "sec" - secondi
- "usec" - microsecondi
- "minuteswest" - minuti ovest di Greenwich
- "dsttime" - tipo di correzione dell'ora legale

gmdate (PHP 3, PHP 4 >= 4.0.0)

Formatta una data/orario GMT/CUT

string **gmdate** (string formato [, int timestamp]) \linebreak

Identica alla funzione date() eccetto per il fatto che l'orario restituito è del Greenwich Mean Time (GMT). Per esempio, quando si è in Finlandia (GMT +0200), la prima linea sotto scrive "Jan 01 1998 00:00:00", mentre la seconda scrive "Dec 31 1997 22:00:00".

Esempio 1. Esempio di gmdate()

```
echo date ("M d Y H:i:s", mktime (0,0,0,1,1,1998));
echo gmdate ("M d Y H:i:s", mktime (0,0,0,1,1,1998));
```

Guarda anche date(), mktime(), gmmktime() e strftime().

gmmktime (PHP 3, PHP 4 >= 4.0.0)

Riceve l'UNIX timestamp per una data GMT

int **gmmktime** (int ora, int minuto, int secondo, int mese, int giorno, int anno [, int is_dst]) \linebreak

Identica alla funzione mktime() eccetto per il paramentro passato, che rappresenta una data GMT.

gmstrftime (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Formatta una data/orario GMT/CUT associandola/o ai parametri locali

string **gmstrftime** (string formato [, int timestamp]) \linebreak

Si comporta allo stesso modo della funzione strftime() eccetto che l'orario restituito è del Greenwich Mean Time (GMT). Ad esempio, quando si è nell' Eastern Standard Time (GMT -0500), la prima linea sotto scrive "Dec 31 1998 20:00:00", mentre la seconda scrive "Jan 01 1999 01:00:00".

Esempio 1. Esempio di gmstrftime()

```
setlocale ('LC_TIME', 'en_US');
echo strftime ("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
echo gmstrftime ("%b %d %Y %H:%M:%S", mktime (20,0,0,12,31,98))."\n";
```

Guarda anche strftime().

localtime (PHP 4 >= 4.0.0)

Riceve l'orario locale

array **localtime** ([int timestamp [, bool is_associative]]) \linebreak

La funzione **localtime()** restituisce un array identico a quello della struttura della chimata della funzione di C. Il primo argomento per **localtime()** è il timestamp, se non esiste viene assegnato di default l'orario attuale. Il secondo argomento di **localtime()** è *is_associative*, se questo è settato a 0 o non sostituito l'array restituirà un regolare array numerico indicizzato. Se l'argomento è settato a 1, **localtime()** è un array associativo contenente tutti gli elementi differenti della struttura restituita dalla funzione chiamata localtime di C. Il nome delle differenti chiavi dell'array associativo sono le seguenti:

- "tm_sec" - secondi
- "tm_min" - minuti
- "tm_hour" - ora
- "tm_mday" - giorno del mese
- "tm_mon" - mese dell'anno, iniziando con 0 per Gennaio
- "tm_year" - Anni dal 1900
- "tm_wday" - Giorno della settimana
- "tm_yday" - Giorno dell'anno
- "tm_isdst" - Se l'ora legale è effettiva

microtime (PHP 3, PHP 4 >= 4.0.0)

Restituisce l'attuale UNIX timestamp con i microsecondi

string **microtime** (void) \linebreak

Restituisce la stringa "msec sec" dove sec è l'attuale orario misurato nel numero di secondi dalla Unix Epoch (0:00:00 January 1, 1970 GMT), e msec è la parte in microsecondi. Questa funzione è disponibile solo su sistemi operativi che supportano la chiamata di sistema gettimeofday().

Entrambi le parti della stringa sono restituite in unità di secondi.

Esempio 1. Esempio di microtime()

```
function getmicrotime(){
    list($usec, $sec) = explode(" ",microtime());
    return ((float)$usec + (float)$sec);
    }

$time_start = getmicrotime();

for ($i=0; $i < 1000; $i++){
    //do nothing, 1000 times
```

```
    }

$time_end = getmicrotime();
$time = $time_end - $time_start;

echo "Did nothing in $time seconds";
```

Vedere anche time().

mktime (PHP 3, PHP 4 >= 4.0.0)

Restituisce la UNIX timestamp per una data

int **mktime** (int hour, int minute, int second, int month, int day, int year [, int is_dst]) \linebreak

Attenzione: Nota lo strano ordine degli argomenti, che differiscono dal normale ordine degli argomenti in una normale chiamata UNIX mktime() e che non si presta bene a far comparire i parametri da destra a sinistra (guarda sotto). E’ un comune errore la confusione di questi argomenti in uno script.

Restituisce la Unix timestamp corrispondente all’argomento dato. Questa timestamp è un intero lungo contenente il numero di secondi tra la Unix Epoch (January 1 1970) e la data e orario specificati.

Gli argomenti possono essere omessi nell’ordine da destra a sinistra; degli argomenti omessi saranno impostati con l’attuale valore accordandolo alla data e orario locale.

`is_dst` può essere impostato su 1 se l’orario è nell’ora legale, 0 altrimenti, o -1 (di default) se è sconosciuta la presenza dell’ora legale o meno. Se è sconosciuto, il PHP proverà ad impostarlo da se. Questo può causare un risultato non aspettato (ma non sbagliato).

> **Nota:** `is_dst` è stato aggiunto nella verisone 3.0.10.

mktime() è usata per fare calcoli tra date e validazioni, come può calcolare automaticamente il corretto valore per un valore fuori dall’intervallo valido. Per esempio, ognuna delle seguenti linee produce la stringa "Jan-01-1998".

Esempio 1. Esempio di mktime()

```
echo date ("M-d-Y", mktime (0,0,0,12,32,1997));
echo date ("M-d-Y", mktime (0,0,0,13,1,1997));
echo date ("M-d-Y", mktime (0,0,0,1,1,1998));
echo date ("M-d-Y", mktime (0,0,0,1,1,98));
```

`Year` può avere sia 2 che 4 cifre, con valori compresi tra 0-69 e 2000-2069 oppure tra 70-99 e 1970-1999 (sui sistemi dove time_t è un intero segnato a 32bit, come sulla maggior parte dei PC di oggi, l'intervallo valido per `year` è tra 1902 e 2037).

L'ultimo giorno del mese dato può essere espresso come il giorno "0" del mese successivo, non come il giorno -1. Entrami i seguenti esempi produrranno la stringa "L'ultimo giorno di Feb 2000 è: 29".

Esempio 2. L'ultimo giorno del mese successivo

```
$lastday = mktime (0,0,0,3,0,2000);
echo strftime ("L'ultimo giorno di Feb 2000 è: %d", $lastday);

$lastday = mktime (0,0,0,4,-31,2000);
echo strftime ("L'ultimo giorno di Feb 2000 è: %d", $lastday);
```

Date con anno, mese e giorno uguali a 0 non sono considerate valide (altrimenti saranno considerate come 30.11.1999, quando hanno uno strano behavior).

Guarda anche date() e time().

strftime (PHP 3, PHP 4 >= 4.0.0)

Formatta una data/orario locale accordandola/o alle impostazioni locali according to locale settings

string **strftime** (string format [, int timestamp]) \linebreak

Restituisce una stringa formattata in accordo con la stringa del formato data usando il parametro dato `timestamp` o l'attuale orario locale se non è stato dato il timestamp. I nomi di mesi e giorni della settimana e le altre stringhe dipendenti dalla lingua rispettano le attuali impostazioni locali con setlocale().

Le seguenti sequenze di caratteri sono utilizzate nella stringa del formato:

- %a - Nome del giorno della settimana abbreviato in accordo con i parametri locali
- %A - Nome completo del giorno della settimana in accordo con i parametri locali
- %b - Nome del mese abbreviato in accordo con i parametri locali
- %B - Nome completo del mese in accordo con i parametri locali
- %c - Rappresentazione preferita di data e orario per le attuali impostazioni locali
- %C - numero del secolo (l'anno diviso 100 e troncato in un intero, intervallo tra 00 e 99)
- %d - giorno del mese come numero decimale (intervallo tra 01 e 31)
- %D - come %m/%d/%y
- %e - giorno del mese come numero decimale, un singolo carattere è preceduto da uno spapzio (intervallo tra ' 1' e '31')

- %g - come %G, ma senza il secolo.
- %G - L’anno a 4 cifre corrispondente al numero di setitmana ISO (vedi %V). Questa ha lo stesso formato e valore di %Y, eccetto che se il numero di settimana ISO appartiene al precedente o prossimo anno, è invece utilizzato l’anno attuale.
- %h - come %b
- %H - ora come numero decimale usando il sistema a 24 ore (intervallo tra 00 e 23)
- %I - ora come numero decimale usando il sistema a 12 ore (intervallo tra 01 e 12)
- %j - giorno dell’anno come numero decimale (intervallo tra 001 e 366)
- %m - mese come numero decimale (intervallo tra 01 e 12)
- %M - minuto come numero decimale
- %n - carattere di nuova linea
- %p - entrambi ‘am’ o ‘pm’ accordati a un valore di tempo dato, o alle stringhe corrispondenti per le impostazioni locali
- %r - orario in notazione a.m. e p.m
- %R - orario nella notazione a 24 ore
- %S - secondi come numero decimale
- %t - Carattere di tabella
- %T - orario attuale, identico a %H:%M:%S
- %u - giorno della settimana come numero decimale [1,7], dove 1 rappresenta il Lunedì

> **Attenzione**
>
> Sun Solaris sembra far iniziare con la Domenica a come 1 sebbe la ISO 9889:1999 (l’attuale standard di C) specifica chiaramente che dovrebbe iniziare dal Lunedì.

- %U - numero della settimana dell’anno in corso come numero decimale, iniziando dalla prima Domenica come primo giorno della prima settimana
- %V - Il numero di settimana ISO 8601:1988 dell’anno attuale come numero decimale, intervallo tra 01 e 53, dove la settimana 1 è la prima settimana che ha almeno 4 giorni dell’attuale anno, e con il Lunedì come primo giorno della settimana. (Utilizza %G o %g per l’anno componente che corrisponde al numero di settimana per il timestamp specificato.)
- %W - numero della settimana dell’attuale anno come numero decimale, partendo con il primo Lunedì come primo giorno della prima settimana
- %w - giorno della settimana come decimale, dove la Domenica è 0
- %x - visualizzazione della data preferita dalle impostazioni del sistema locale senza orario
- %X - visualizzazione dell’orario preferito dalle impostazioni del sistema locale senza data
- %y - anno come numero decimale senza secolo (intervallo tra 00 e 99)
- %Y - anno come numero decimale incluso il secolo
- %Z - fuso orario o abbreviazione
- %% - il carattere ‘%’

> **Nota:** Non tutte le sequenze di caratteri potrebbero essere supportate dalla tua libreria locale di C, in tal caso la funzione **strftime()** non sarà supportata dal PHP. Questo significa che %T e %D non funzioneranno sotto Windows.

Esempio 1. Esempio di strftime()

```
setlocale (LC_TIME, "C");
print (strftime ("%A in Finlandese è "));
setlocale (LC_TIME, "fi_FI");
print (strftime ("%A, in Francese "));
setlocale (LC_TIME, "fr_FR");
print (strftime ("%A e in Italiano "));
setlocale (LC_TIME, "it_IT");
print (strftime ("%A.\n"));
```

Questo esempio funziona se hai le ripettive impostazioni di lingua installate nel tuo sistema.

Guarda anche setlocale() e mktime() e le specifiche dell' Open Group per **strftime()** (http://www.opengroup.org/onlinepubs/7908799/xsh/strftime.html).

strtotime (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Analizza le descrizioni testuali di datetime in Inglese nell'UNIX timestamp

int **strtotime** (string time [, int now]) \linebreak

La funzione aspetta di avere una stringa contenente un formato di data in Inglese e proverà a passare questo formato all'UNIX timestamp relativo al timestamp dato in *`now`*, o l'attuale orario se non è stato passato il parametro. Sul fallimento, è restituito `-1`.

Perché **strtotime()** si comporti in accordo con la sintassi della data GNU, dai uno sguardo alla pagina del manuale GNU intitolata Date Input Formats (http://www.gnu.org/manual/tar-1.12/html_chapter/tar_7.html). Descrive se c'è una sinstassi valida per il parametro *`time`*.

Esempio 1. strtotime() - esempi

```
echo strtotime ("now"), "\n";
echo strtotime ("10 September 2000"), "\n";
echo strtotime ("+1 day"), "\n";
echo strtotime ("+1 week"), "\n";
echo strtotime ("+1 week 2 days 4 hours 2 seconds"), "\n";
echo strtotime ("next Thursday"), "\n";
echo strtotime ("last Monday"), "\n";
```

Esempio 2. Controllo per il fallimento

```
$str = ’Not Good’;
if (($timestamp = strtotime($str)) === -1) {
    echo "The string ($str) is bogus";
} else {
    echo "$str == ". date(’l dS of F Y h:i:s A’,$timestamp);
}
```

> **Nota:** L’intervallo valido di un timestamp è normalmente da Fri, 13 Dec 1901 20:45:54 GMT a Tue, 19 Jan 2038 03:14:07 GMT. (Queste sono le date ce corrispondono al minimo e al massimo valore per un intero segnato a 32 bit.)

time (PHP 3, PHP 4 >= 4.0.0)

Restituisce l’attuale UNIX timestamp

int **time** (void) \linebreak

Restituisce l’attuale data e orario misurata in numero di secondi dalla Unix Epoch (January 1 1970 00:00:00 GMT).

Guarda anche date().

XIX. Funzioni dBase

Queste funzioni consentono di accedere ai records memorizzati in dBase-format (dbf) database. Per poter usare queste funzioni, si deve prima compilare il PHP con l'opzione `--enable-dbase` option.

Non è previsto il supporto per indici o campi memo. Manca anche il supporto per il locking. E' probabile che due processi concorrenti da parte del webserver sugli stessi file dBase, finiscano con il rovinare il Database.

I files dBase sono semplici files sequenziali o records a lunghezza fissa. I record sono appesi alla fine del file e quelli cancellati sono conservati fino alla chiamata della funzione dbase_pack().

Vi raccomandiamo di non usare i files dBase come database di lavoro. Scegliete piuttosto qualsiasi reale SQL server; MySQL o Postgres sono scelte comuni con PHP. Il supporto dBase è presente per permettervi di importare ed esportare dati da e verso il vostro web database, perchè il formato del file è comunemente ben interpretato dai fogli elettronici e dagli organizers di Windows.

dbase_add_record (PHP 3, PHP 4 >= 4.0.0)

Aggiunge un record ad un database dBase

bool **dbase_add_record** (int dbase_identifier, array record) \linebreak

Aggiunge i dati nel *record* al database. Se il numero di items nel record non è uguale al numero di campi nel database, l'operazione fallirà e sarà restituito FALSE.

dbase_close (PHP 3, PHP 4 >= 4.0.0)

Chiude un database dBase

bool **dbase_close** (int dbase_identifier) \linebreak

Chiude il database associato con il *dbase_identifier*.

dbase_create (PHP 3, PHP 4 >= 4.0.0)

Crea un database dBase

int **dbase_create** (string filename, array fields) \linebreak

Il parametro *fields* è un array di arrays, ciascun array descrive il formato di un campo nel database. Ogni campo consiste di un nome, un carattere che indica il tipo di campo, una lunghezza, e una precisione.

I tipi di campo disponibili sono:

L

Boolean. Questi non hanno una lunghezza o una precisione.

M

Memo. (Nota che non sono supportati da PHP.) Questi non hanno una lunghezza o una precisione.

D

Date (memorizzate nel formato YYYYMMDD). Questi non hanno una lunghezza o una precisione.

N

Number. Questi hanno sia una lunghezza sia una precisione (il numero di decimali).

C

String.

Se il database è creato con successo, è restituito un dbase_identifier, altrimenti è restituito FALSE.

Esempio 1. Creare un file di database dBase

```
// "database" name
$dbname = "/tmp/test.dbf";

// database "definition"
$def =
    array(
        array("date",     "D"),
        array("name",     "C",  50),
        array("age",      "N",   3, 0),
        array("email",    "C", 128),
        array("ismember", "L")
    );

// creation
if (!dbase_create($dbname, $def))
    print "<strong>Error!</strong>";
```

dbase_delete_record (PHP 3, PHP 4 >= 4.0.0)

Cancella un record da un database dBase

bool **dbase_delete_record** (int dbase_identifier, int record) \linebreak

Marca il *record* da cancellare dal database. Per rimuovere il record dal database, è necessario chiamare la funzione dbase_pack().

dbase_get_record (PHP 3, PHP 4 >= 4.0.0)

Estrae un record da un database dBase

array **dbase_get_record** (int dbase_identifier, int record) \linebreak

Restituisce i dati da *record* in un array. L'array è indicizzato partendo da 0, e include un membro associativo chiamato 'deleted' che è settato a 1 se il record è stato marcato per la cancellazione (vedere dbase_delete_record()).

Ogni campo è convertito all'appropriato tipo PHP, Fanno eccezione:

- Le date, che sono lasciate come stringhe.
- Gli interi che avrebbero causato un overflow (> 32 bits) sono restituiti come stringhe.

dbase_get_record_with_names (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Estrae un record da un database dBase come un array associativo

array **dbase_get_record_with_names** (int dbase_identifier, int record) \linebreak

Restituisce i dati da `record` in un array associativo. L'array include anche un membro associativo chiamato 'deleted' che è settato a 1 se il record è stato marcato per la cancellazione (vedere dbase_delete_record()).

Ogni campo è convertito all'appropriato tipo PHP, Fanno eccezione:

- Le date, che sono lasciate come stringhe.
- Gli interi che avrebbero causato un overflow (> 32 bits) sono restituiti come stringhe.

dbase_numfields (PHP 3, PHP 4 >= 4.0.0)

Restituisce il numero di campi in un database dBase.

int **dbase_numfields** (int dbase_identifier) \linebreak

Restituisce il numero di campi (colonne) nel databaase specificato. I numeri dei campi sono compresi tra 0 e dbase_numfields($db)-1, mentre i numeri dei records sono compresi tra 1 e dbase_numrecords($db).

Esempio 1. Usare dbase_numfields()

```
$rec = dbase_get_record($db, $recno);
$nf  = dbase_numfields($db);
for ($i=0; $i < $nf; $i++) {
    print $rec[$i]."<br>\n";
}
```

dbase_numrecords (PHP 3, PHP 4 >= 4.0.0)

Restituisce il numero di records in un database dBase.

int **dbase_numrecords** (int dbase_identifier) \linebreak

Restituisce il numero di records (righe) nel database specificato. I numeri dei records sono compresi tra 1 e dbase_numrecords($db), mentre i numeri dei campi sono compresi tra 0 e dbase_numfields($db)-1.

dbase_open (PHP 3, PHP 4 >= 4.0.0)

Apre un database dBase

int **dbase_open** (string filename, int flags) \linebreak

I flags corrispondono a quelli per la chiamata all' open() system. (Tipicamente 0 significa sola lettura, 1 sola scrittura e 2 lettura e scrittura.)

Restituisce un dbase_identifier per il database aperto, o `FALSE` se il database non può essere aperto

Nota: Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali si sta andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

dbase_pack (PHP 3, PHP 4 >= 4.0.0)

Stabilizza un database dBase

bool **dbase_pack** (int dbase_identifier) \linebreak

Stabilizza il database specificato (cancellando permanentemente tutti i records marcati per la cancellazione usando dbase_delete_record()).

dbase_replace_record (PHP 3>= 3.0.11, PHP 4 >= 4.0.0)

Sostituisce un record in un database dBase

bool **dbase_replace_record** (int dbase_identifier, array record, int dbase_record_number) \linebreak

Sostituisce i dati associati con il record *`record_number`* con i dati nel *`record`* nel database. Se il numero di items nel record non è uguale al numero di campi nel database, l'operazione fallirà e sarà restituito `FALSE`.

`dbase_record_number` è un integer che va da 1 al numero di records nel database (come restituito da dbase_numrecords()).

XX. Funzioni DBM

Questa funzioni consentono lo storage di records memorizzati in un dbm-style database. Questo tipo di database (supportato da Berkeley DB, GDBM, e qualche libreria di sistema, così come una built-in flatfile library) memorizza coppie key/value (al contrario dei full-blown records supportati dai database relazionali).

Esempio 1. Esempio DBM

```
$dbm = dbmopen ("lastseen", "w");
if (dbmexists ($dbm, $userid)) {
    $last_seen = dbmfetch ($dbm, $userid);
} else {
    dbminsert ($dbm, $userid, time());
}
do_stuff();
dbmreplace ($dbm, $userid, time());
dbmclose ($dbm);
```

dblist (PHP 3, PHP 4 >= 4.0.0)

Descrive la libreria DBM-compatibile in uso.

string **dblist** (void) \linebreak

dbmclose (PHP 3, PHP 4 >= 4.0.0)

Chiude un database dbm

bool **dbmclose** (int dbm_identifier) \linebreak

Sblocca e chiude il database specificato.

dbmdelete (PHP 3, PHP 4 >= 4.0.0)

Cancella il valore per una chiave da un database DBM

bool **dbmdelete** (int dbm_identifier, string key) \linebreak

Cancella il valore per *`key`* nel database.

Restituisce `FALSE` se la chiave non esisteva nel database.

dbmexists (PHP 3, PHP 4 >= 4.0.0)

Dice se esiste un valore per una chiave in un database DBM

bool **dbmexists** (int dbm_identifier, string key) \linebreak

Restituisce `TRUE` se c'è un valore associato con la *`key`*.

dbmfetch (PHP 3, PHP 4 >= 4.0.0)

Estrae un valore per una chiave da un database DBM

string **dbmfetch** (int dbm_identifier, string key) \linebreak

Restituisce il valore associato con *`key`*.

dbmfirstkey (PHP 3, PHP 4 >= 4.0.0)

Recupera la prima chiave da un database DBM

string **dbmfirstkey** (int dbm_identifier) \linebreak

Restituisce la prima chiave nel database. Da notare che nessun ordine particolare è garantito dal momento che il database potrebbe essere stato costruito usando una hash-table, che non garantisce nessun ordine.

dbminsert (PHP 3, PHP 4 >= 4.0.0)

Inserisce un valore per una chiave in un database DBM

int **dbminsert** (int dbm_identifier, string key, string value) \linebreak

Aggiunge il valore al database con la chiave specificata.

Restituisce -1 se il database è stato aperto in modalità read-only (sola lettura), 0 se l'inserimento è andato a buon fine, e 1 se la chiave specificata già esiste. (Per sostituire il valore, usa dbmreplace().)

dbmnextkey (PHP 3, PHP 4 >= 4.0.0)

Recupera la chiave successiva da un database DBM

string **dbmnextkey** (int dbm_identifier, string key) \linebreak

Restituisce la chiave successiva dopo `key`. Chiamando la funzione dbmfirstkey() seguita da successive chiamate alla funzione **dbmnextkey()** è possibile visionare ciascuna coppia key/value nel database DBM. Per esempio:

Esempio 1. Esaminare ogni coppia key/value in un database DBM

```
$key = dbmfirstkey ($dbm_id);
while ($key) {
    echo "$key = " . dbmfetch ($dbm_id, $key) . "\n";
    $key = dbmnextkey ($dbm_id, $key);
}
```

dbmopen (PHP 3, PHP 4 >= 4.0.0)

Apre un database DBM

int **dbmopen** (string filename, string flags) \linebreak

Il primo argomento è il nome (con il percorso completo) del file DBM da aprire e il secondo è l' open mode del file che può essere "r", "n", "c" or "w" per sola lettura, nuovo (implica lettura-scrittura, e molto probabilmente troncherà un database esistente con lo stesso nome), crea

(implica lettura-scrittura, e non troncherà un database esistente con lo stesso nome) e lettura-scrittura rispettivamente.

Restituisce un identifier da passare alle altre funzioni DBM in caso di successo, o `FALSE` se fallisce.

Se è usato il supporto NDBM, NDBM creerà filename.dir e filename.pag files. GDBM usa solo un file, in quanto fa l' internal flat-file support, e il Berkeley DB crea un `filename.db` file. Da notare che PHP fa il suo file locking in aggiunta a quello che eventualmente potrebbe fare la stessa libreria DBM. PHP non cancella i files `.lck` che crea. Semplicemente usa questi files come fixed inodes su cui fare il file locking. Per maggiori informazioni sui files DBM, guarda le pagine del tuo manuale UNIX, o scarica GNU's GDBM (ftp://ftp.gnu.org/pub/gnu/gdbm/).

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali si sta andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

dbmreplace (PHP 3, PHP 4 >= 4.0.0)

Sostituisce il valore per una chiave in un database DBM

int **dbmreplace** (int dbm_identifier, string key, string value) \linebreak

Sostituisce il valore per la chiave specificata nel database.

Aggiungerà inoltre la chiave al database se la stessa non esisteva già.

XXI. dbx functions

Introduzione

The dbx module is a database abstraction layer (db 'X', where 'X' is a supported database). The dbx functions allow you to access all supported databases using a single calling convention. The dbx-functions themselves do not interface directly to the databases, but interface to the modules that are used to support these databases.

Requisiti

To be able to use a database with the dbx-module, the module must be either linked or loaded into PHP, and the database module must be supported by the dbx-module. Currently, following databases are supported, but others will follow (soon, I hope :-):

- FrontBase (available from PHP 4.1.0).
- Microsoft SQL Server
- MySQL
- ODBC
- PostgreSQL
- Sybase-CT (available from PHP 4.2.0).

Documentation for adding additional database support to dbx can be found at http://www.guidance.nl/php/dbx/doc/.

Istallazione

In order to have these functions available, you must compile PHP with dbx support by using the `--enable-dbx` option and all options for the databases that will be used, e.g. for MySQL you must also specify `--with-mysql=[DIR]`. To get other supported databases to work with the dbx-module refer to their specific documentation.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

There are two resource types used in the dbx module. The first one is the link-object for a database

connection, the second a result-object which helds the result of a query.

Costanti Predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata

compilata nel PHP o se è stata caricata dinamicamente a runtime.

`DBX_MYSQL` (integer)

`DBX_ODBC` (integer)

`DBX_PGSQL` (integer)

`DBX_MSSQL` (integer)

`DBX_FBSQL` (integer)

`DBX_OCI8` (integer)

`DBX_SYBASECT` (integer)

`DBX_PERSISTENT` (integer)

`DBX_RESULT_INFO` (integer)

`DBX_RESULT_INDEX` (integer)

`DBX_RESULT_ASSOC` (integer)

`DBX_CMP_NATIVE` (integer)

`DBX_CMP_TEXT` (integer)

`DBX_CMP_NUMBER` (integer)

`DBX_CMP_ASC` (integer)

`DBX_CMP_DESC` (integer)

dbx_close (PHP 4 >= 4.0.6)

Close an open connection/database

bool **dbx_close** (object link_identifier) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

Esempio 1. dbx_close() example

```
<?php
$link = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Could not connect");

print("Connected successfully");
dbx_close($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_connect().

dbx_compare (PHP 4 >= 4.1.0)

Compare two rows for sorting purposes

int **dbx_compare** (array row_a, array row_b, string column_key [, int flags]) \linebreak

dbx_compare() returns `0` if the `row_a[$column_key]` is equal to `row_b[$column_key]`, and `1` or `-1` if the former is greater or is smaller than the latter one, respectively, or vice versa if the *flag* is set to `DBX_CMP_DESC`. **dbx_compare()** is a helper function for dbx_sort() to ease the make and use of the custom sorting function.

The *flags* can be set to specify comparison direction:

- `DBX_CMP_ASC` - ascending order
- `DBX_CMP_DESC` - descending order

and the preferred comparison type:

- `DBX_CMP_NATIVE` - no type conversion
- `DBX_CMP_TEXT` - compare items as strings
- `DBX_CMP_NUMBER` - compare items numerically

One of the direction and one of the type constant can be combined with bitwise OR operator (|). The default value for the *flags* parameter is `DBX_CMP_ASC` | `DBX_CMP_NATIVE`.

Esempio 1. dbx_compare() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect (DBX_ODBC, "", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query ($link, "SELECT id, parentid, description FROM table OR-
DER BY id");
    // data in $result is now ordered by id

dbx_sort ($result, "user_re_order");
    // date in $result is now ordered by parentid (descending), then by id

dbx_close ($link);
?>
```

See also dbx_sort().

dbx_connect (PHP 4 >= 4.0.6)

Open a connection/database

object **dbx_connect** (mixed module, string host, string database, string username, string password [, int persistent]) \linebreak

dbx_connect() returns an object on success, `FALSE` on error. If a connection has been made but the database could not be selected, the connection is closed and `FALSE` is returned. The *`persistent`* parameter can be set to `DBX_PERSISTENT`, if so, a persistent connection will be created.

The *`module`* parameter can be either a string or a constant, though the latter form is preferred. The possible values are given below, but keep in mind that they only work if the module is actually loaded.

- `DBX_MYSQL` or "mysql"
- `DBX_ODBC` or "odbc"
- `DBX_PGSQL` or "pgsql"
- `DBX_MSSQL` or "mssql"
- `DBX_FBSQL` or "fbsql" (available from PHP 4.1.0)
- `DBX_SYBASECT` or "sybase_ct" (available from PHP 4.2.0)

The *host*, *database*, *username* and *password* parameters are expected, but not always used depending on the connect functions for the abstracted module.

The returned `object` has three properties:

database

It is the name of the currently selected database.

handle

It is a valid handle for the connected database, and as such it can be used in module-specific functions (if required).

```
$link = dbx_connect (DBX_MYSQL, "localhost", "db", "username", "password");
mysql_close ($link->handle); // dbx_close($link) would be better here
```

module

It is used internally by dbx only, and is actually the module number mentioned above.

Esempio 1. dbx_connect() example

```
<?php
$link = dbx_connect (DBX_ODBC, "", "db", "username", "password", DBX_PERSISTENT)
    or die ("Could not connect");

print ("Connected successfully");
dbx_close ($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_close().

dbx_error (PHP 4 >= 4.0.6)

Report the error message of the latest function call in the module (not just in the connection)

string **dbx_error** (object link_identifier) \linebreak

dbx_error() returns a string containing the error message from the last function call of the abstracted module (e.g. mysql module). If there are multiple connections in the same module, just

the last error is given. If there are connections on different modules, the latest error is returned for the module specified by the *link_identifier* parameter.

Esempio 1. dbx_error() example

```
<?php
$link   = dbx_connect(DBX_MYSQL, "localhost", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query($link, "select id from non_existing_table");
if ( $result == 0 ) {
    echo dbx_error ($link);
}
dbx_close ($link);
?>
```

Nota: Always refer to the module-specific documentation as well.

The error message for Microsoft SQL Server is actually the result of the mssql_get_last_message() function.

dbx_query (PHP 4 >= 4.0.6)

Send a query and fetch all results (if any)

object **dbx_query** (object link_identifier, string sql_statement [, long flags]) \linebreak

dbx_query() returns an object or `1` on success, and `0` on failure. The result object is returned only if the query given in *sql_statement* produces a result set.

Esempio 1. How to handle the returned value

```
<?php
$link   = dbx_connect(DBX_ODBC, "", "db", "username", "password")
    or die("Could not connect");

$result = dbx_query($link, 'SELECT id, parentid, description FROM table');

if ( is_object($result) ) {
    // ... do some stuff here, see detailed examples below ...
    // first, print out field names and types
    // then, draw a table filled with the returned field values
}
else if ( $result == 1 ) {
    echo("Query executed successfully, but no result set returned");
}
else {
    exit("Query failed");
```

```
}

dbx_close($link);
?>
```

The *flags* parameter is used to control the amount of information that is returned. It may be any combination of the following constants with the bitwise OR operator (|):

DBX_RESULT_INDEX

It is *always* set, that is, the returned object has a data property which is a 2 dimensional array indexed numerically. For example, in the expression `data[2][3]` `2` stands for the row (or record) number and `3` stands for the column (or field) number. The first row and column are indexed at 0.

If `DBX_RESULT_ASSOC` is also specified, the returning object contains the information related to `DBX_RESULT_INFO` too, even if it was not specified.

DBX_RESULT_INFO

It provides info about columns, such as field names and field types.

DBX_RESULT_ASSOC

It effects that the field values can be accessed with the respective column names used as keys to the returned object's data property.

Associated results are actually references to the numerically indexed data, so modifying `data[0][0]` causes that `data[0]['field_name_for_first_column']` is modified as well.

Note that `DBX_RESULT_INDEX` is always used, regardless of the actual value of *flags* parameter. This means that the following combinations is effective only:

- `DBX_RESULT_INDEX`
- `DBX_RESULT_INDEX` | `DBX_RESULT_INFO`
- `DBX_RESULT_INDEX` | `DBX_RESULT_INFO` | `DBX_RESULT_ASSOC` - this is the default, if *flags* is not specified.

The returing `object` has four or five properties depending on *flags*:

handle

It is a valid handle for the connected database, and as such it can be used in module specific functions (if required).

```
$result = dbx_query ($link, "SELECT id FROM table");
mysql_field_len ($result->handle, 0);
```

cols and rows

These contain the number of columns (or fields) and rows (or records) respectively.

```
$result = dbx_query ($link, 'SELECT id FROM table');
echo $result->rows; // number of records
echo $result->cols; // number of fields
```

info (optional)

It is returned only if either `DBX_RESULT_INFO` or `DBX_RESULT_ASSOC` is specified in the *`flags`* parameter. It is a 2 dimensional array, that has two named rows (`name` and `type`) to retrieve column information.

Esempio 2. lists each field's name and type

```
$result = dbx_query ($link, 'SELECT id FROM table',
                     DBX_RESULT_INDEX | DBX_RESULT_INFO);

for ($i = 0; $i < $result->cols; $i++ ) {
    echo $result->info['name'][$i] . "\n";
    echo $result->info['type'][$i] . "\n";
}
```

data

This property contains the actual resulting data, possibly associated with column names as well depending on *`flags`*. If `DBX_RESULT_ASSOC` is set, it is possible to use `$result->data[2]["field_name"]`.

Esempio 3. outputs the content of data property into HTML table

```
$result = dbx_query ($link, 'SELECT id, parentid, description FROM table');

echo "<table>\n";
foreach ( $result->data as $row ) {
    echo "<tr>\n";
    foreach ( $row as $field ) {
        echo "<td>$field</td>";
    }
    echo "</tr>\n";
}
echo "</table>\n";
```

Nota: Always refer to the module-specific documentation as well.

See also: dbx_connect().

dbx_sort (PHP 4 >= 4.0.6)

Sort a result from a dbx_query by a custom sort function

bool **dbx_sort** (object result, string user_compare_function) \linebreak

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

Nota: It is always better to use ORDER BY SQL clause instead of **dbx_sort()**, if possible.

Esempio 1. dbx_sort() example

```
<?php
function user_re_order ($a, $b) {
    $rv = dbx_compare ($a, $b, "parentid", DBX_CMP_DESC);
    if ( !$rv ) {
        $rv = dbx_compare ($a, $b, "id", DBX_CMP_NUMBER);
    }
    return $rv;
}

$link   = dbx_connect (DBX_ODBC, "", "db", "username", "password")
    or die ("Could not connect");

$result = dbx_query ($link, "SELECT id, parentid, description FROM tbl ORDER BY id");
    // data in $result is now ordered by id

dbx_sort ($result, "user_re_order");
    // data in $result is now ordered by parentid (descending), then by id

dbx_close ($link);
?>
```

See also dbx_compare().

XXII. DB++ Functions

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Introduzione

db++, made by the german company Concept asa (http://www.concept-asa.de/), is a relational database system with high performance and low memory and disk usage in mind. While providing SQL as an additional language interface it is not really a SQL database in the first place but provides its own AQL query language which is much more influenced by the relational algebra then SQL is.

Concept asa always had an interest in supporting open source languages, db++ has had Perl and Tcl call interfaces for years now and uses Tcl as its internal stored procedure language.

Requisiti

This extension relies on external client libraries so you have to have a db++ client installed on the system you want to use this extension on.

Concept asa (http://www.concept-asa.de/) provides db++ Demo versions (http://www.concept-asa.de/down-eng.html) and documentation (http://www.concept-asa.de/downloads/doc-eng.tar.gz) for Linux, some other UNIX versions. There is also a Windows version of db++, but this extension doesn't support it (yet).

Istallazione

In order to build this extension yourself you need the db++ client libraries and header files to be installed on your system (these are included in the db++ installation archives by default). You have to run **configure** with option `--with-dbplus` to build this extension.

configure looks for the client libraries and header files under the default paths `/usr/dbplus`, `/usr/local/dbplus` and `/opt/dblus`. If you have installed db++ in a different place you have add the installation path to the **configure** option like this:

`--with-dbplus=/your/installation/path`.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

dbplus_relation

Most db++ functions operate on or return *dbplus_relation* resources. A *dbplus_relation* is a handle to a stored relation or a relation generated as the result of a query.

Costanti Predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata compilata nel PHP o se è stata caricata dinamicamente a runtime.

db++ error codes

Tabella 1. DB++ Error Codes

PHP Constant	db++ constant	meaning
`DBPLUS_ERR_NOERR` (integer)	ERR_NOERR	Null error condition
`DBPLUS_ERR_DUPLICATE` (integer)	ERR_DUPLICATE	Tried to insert a duplicate tuple
`DBPLUS_ERR_EOSCAN` (integer)	ERR_EOSCAN	End of scan from rget()
`DBPLUS_ERR_EMPTY` (integer)	ERR_EMPTY	Relation is empty (server)
`DBPLUS_ERR_CLOSE` (integer)	ERR_CLOSE	The server can't close
`DBPLUS_ERR_WLOCKED` (integer)	ERR_WLOCKED	The record is write locked
`DBPLUS_ERR_LOCKED` (integer)	ERR_LOCKED	Relation was already locked
`DBPLUS_ERR_NOLOCK` (integer)	ERR_NOLOCK	Relation cannot be locked
`DBPLUS_ERR_READ` (integer)	ERR_READ	Read error on relation
`DBPLUS_ERR_WRITE` (integer)	ERR_WRITE	Write error on relation
`DBPLUS_ERR_CREATE` (integer)	ERR_CREATE	Create() system call failed
`DBPLUS_ERR_LSEEK` (integer)	ERR_LSEEK	Lseek() system call failed

PHP Constant	db++ constant	meaning
DBPLUS_ERR_LENGTH (integer)	ERR_LENGTH	Tuple exceeds maximum length
DBPLUS_ERR_OPEN (integer)	ERR_OPEN	Open() system call failed
DBPLUS_ERR_WOPEN (integer)	ERR_WOPEN	Relation already opened for writing
DBPLUS_ERR_MAGIC (integer)	ERR_MAGIC	File is not a relation
DBPLUS_ERR_VERSION (integer)	ERR_VERSION	File is a very old relation
DBPLUS_ERR_PGSIZE (integer)	ERR_PGSIZE	Relation uses a different page size
DBPLUS_ERR_CRC (integer)	ERR_CRC	Invalid crc in the superpage
DBPLUS_ERR_PIPE (integer)	ERR_PIPE	Piped relation requires lseek()
DBPLUS_ERR_NIDX (integer)	ERR_NIDX	Too many secondary indices
DBPLUS_ERR_MALLOC (integer)	ERR_MALLOC	Malloc() call failed
DBPLUS_ERR_NUSERS (integer)	ERR_NUSERS	Error use of max users
DBPLUS_ERR_PREEXIT (integer)	ERR_PREEXIT	Caused by invalid usage
DBPLUS_ERR_ONTRAP (integer)	ERR_ONTRAP	Caused by a signal
DBPLUS_ERR_PREPROC (integer)	ERR_PREPROC	Error in the preprocessor
DBPLUS_ERR_DBPARSE (integer)	ERR_DBPARSE	Error in the parser
DBPLUS_ERR_DBRUNERR (integer)	ERR_DBRUNERR	Run error in db
DBPLUS_ERR_DBPREEXIT (integer)	ERR_DBPREEXIT	Exit condition caused by prexit() * procedure
DBPLUS_ERR_WAIT (integer)	ERR_WAIT	Wait a little (Simple only)
DBPLUS_ERR_CORRUPT_TUPLE (integer)	ERR_CORRUPT_TUPLE	A client sent a corrupt tuple
DBPLUS_ERR_WARNING0 (integer)	ERR_WARNING0	The Simple routines encountered a non fatal error which was corrected
DBPLUS_ERR_PANIC (integer)	ERR_PANIC	The server should not really die but after a disaster send ERR_PANIC to all its clients
DBPLUS_ERR_FIFO (integer)	ERR_FIFO	Can’t create a fifo
DBPLUS_ERR_PERM (integer)	ERR_PERM	Permission denied
DBPLUS_ERR_TCL (integer)	ERR_TCL	TCL_error
DBPLUS_ERR_RESTRICTED (integer)	ERR_RESTRICTED	Only two users
DBPLUS_ERR_USER (integer)	ERR_USER	An error in the use of the library by an application programmer

PHP Constant	db++ constant	meaning
DBPLUS_ERR_UNKNOWN (integer)	ERR_UNKNOWN	

dbplus_add (PHP 4 >= 4.1.0)

Add a tuple to a relation

int **dbplus_add** (resource relation, array tuple) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

This function will add a tuple to a relation. The *tuple* data is an array of attribute/value pairs to be inserted into the given *relation*. After successful execution the *tuple* array will contain the complete data of the newly created tuple, including all implicitly set domain fields like sequences.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_aql (PHP 4 >= 4.1.0)

Perform AQL query

resource **dbplus_aql** (string query [, string server [, string dbpath]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_aql() will execute an AQL *query* on the given *server* and *dbpath*.

On success it will return a relation handle. The result data may be fetched from this relation by calling dbplus_next() and **dbplus_current()**. Other relation access functions will not work on a result relation.

Further information on the AQL A... Query Language is provided in the original db++ manual.

dbplus_chdir (PHP 4 >= 4.1.0)

Get/Set database virtual current directory

string **dbplus_chdir** ([string newdir]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_chdir() will change the virtual current directory where relation files will be looked for by dbplus_open(). **dbplus_chdir()** will return the absolute path of the current directory. Calling **dbplus_chdir()** without giving any *newdir* may be used to query the current working directory.

dbplus_close (PHP 4 >= 4.1.0)

Close a relation

int **dbplus_close** (resource relation) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Calling **dbplus_close()** will close a relation previously opened by dbplus_open().

dbplus_curr (PHP 4 >= 4.1.0)

Get current tuple from relation

int **dbplus_curr** (resource relation, array tuple) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_curr() will read the data for the current tuple for the given *relation* and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_errcode (PHP 4 >= 4.1.0)

Get error string for given errorcode or last error

string **dbplus_errcode** (int errno) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_errcode() returns a cleartext error string for the error code passed as *errno* of for the result code of the last db++ operation if no parameter is given.

dbplus_errno (PHP 4 >= 4.1.0)

Get error code for last operation

int **dbplus_errno** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_errno() will return the error code returned by the last db++ operation.

See also dbplus_errcode().

dbplus_find (PHP 4 >= 4.1.0)

Set a constraint on a relation

int **dbplus_find** (resource relation, array constraints, mixed tuple) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_find() will place a constraint on the given relation. Further calls to functions like dbplus_curr() or dbplus_next() will only return tuples matching the given constraints.

Constraints are triplets of strings containing of a domain name, a comparison operator and a comparison value. The `constraints` parameter array may consist of a collection of string arrays, each of which contains a domain, an operator and a value, or of a single string array containing a multiple of three elements.

The comparison operator may be one of the following strings: '==', '>', '>=', '<', '<=', '!=', '~' for a regular expression match and 'BAND' or 'BOR' for bitwise operations.

See also dbplus_unselect().

dbplus_first (PHP 4 >= 4.1.0)

Get first tuple from relation

int **dbplus_first** (resource relation, array tuple) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_curr() will read the data for the first tuple for the given `relation`, make it the current tuple and pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_curr(), dbplus_prev(), dbplus_next(), and dbplus_last().

dbplus_flush (PHP 4 >= 4.1.0)

Flush all changes made on a relation

int **dbplus_flush** (resource relation) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_flush() will write all changes applied to `relation` since the last flush to disk.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_freealllocks (PHP 4 >= 4.1.0)

Free all locks held by this client

int **dbplus_freealllocks** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_freeaalllocks() will free all tuple locks held by this client.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freerlocks().

dbplus_freelock (PHP 4 >= 4.1.0)

Release write lock on tuple

int **dbplus_freelock** (resource relation, string tname) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_freelock() will release a write lock on the given `tuple` previously obtained by dbplus_getlock().

See also dbplus_getlock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_freerlocks (PHP 4 >= 4.1.0)

Free all tuple locks on given relation

int **dbplus_freerlocks** (resource relation) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_freerlocks() will free all tuple locks held on the given `relation`.

See also dbplus_getlock(), dbplus_freelock(), and dbplus_freealllocks().

dbplus_getlock (PHP 4 >= 4.1.0)

Get a write lock on a tuple

int **dbplus_getlock** (resource relation, string tname) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_getlock() will request a write lock on the specified *tuple*. It will return zero on success or a non-zero error code, especially DBPLUS_ERR_WLOCKED, on failure.

See also dbplus_freelock(), dbplus_freerlocks(), and dbplus_freealllocks().

dbplus_getunique (PHP 4 >= 4.1.0)

Get a id number unique to a relation

int **dbplus_getunique** (resource relation, int uniqueid) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_getunique() will obtain a number guaranteed to be unique for the given *relation* and will pass it back in the variable given as *uniqueid*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

dbplus_info (PHP 4 >= 4.1.0)

???

int **dbplus_info** (resource relation, string key, array) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Not implemented yet.

dbplus_last (PHP 4 >= 4.1.0)

Get last tuple from relation

int **dbplus_last** (resource relation, array tuple) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_curr() will read the data for the last tuple for the given `relation`, make it the current tuple and pass it back as an associative array in `tuple`.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_next().

dbplus_lockrel (unknown)

Request write lock on relation

int **dbplus_lockrel** (resource relation) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_lockrel() will request a write lock on the given relation. Other clients may still query the relation, but can't alter it while it is locked.

dbplus_next (PHP 4 >= 4.1.0)

Get next tuple from relation

int **dbplus_next** (resource relation, array) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_curr() will read the data for the next tuple for the given *relation*, will make it the current tuple and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_prev(), and dbplus_last().

dbplus_open (PHP 4 >= 4.1.0)

Open relation file

resource **dbplus_open** (string name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

The relation file *name* will be opened. *name* can be either a file name or a relative or absolute path name. This will be mapped in any case to an absolute relation file path on a specific host machine and server.

On success a relation file resource (cursor) is returned which must be used in any subsequent commands referencing the relation. Failure leads to a zero return value, the actual error code may be asked for by calling dbplus_errno().

dbplus_prev (PHP 4 >= 4.1.0)

Get previous tuple from relation

int **dbplus_prev** (resource relation, array tuple) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_curr() will read the data for the next tuple for the given *relation*, will make it the current tuple and will pass it back as an associative array in *tuple*.

The function will return zero (aka. DBPLUS_ERR_NOERR) on success or a db++ error code on failure. See dbplus_errcode() or the introduction to this chapter for more information on db++ error codes.

See also dbplus_first(), dbplus_curr(), dbplus_next(), and dbplus_last().

dbplus_rchperm (PHP 4 >= 4.1.0)

Change relation permissions

int **dbplus_rchperm** (resource relation, int mask, string user, string group) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rchperm() will change access permissions as specified by *mask*, *user* and *group*. The values for these are operating system specific.

dbplus_rcreate (PHP 4 >= 4.1.0)

Creates a new DB++ relation

resource **dbplus_rcreate** (string name, mixed domlist [, boolean overwrite]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rcreate() will create a new relation named *name*. An existing relation by the same name will only be overwritten if the relation is currently not in use and *overwrite* is set to TRUE.

domlist should contain the domain specification for the new relation within an array of domain description strings. (**dbplus_rcreate()** will also accept a string with space delimited domain

description strings, but it is recommended to use an array). A domain description string consists of a domain name unique to this relation, a slash and a type specification character. See the db++ documentation, especially the dbcreate(1) manpage, for a description of available type specifiers and their meanings.

dbplus_rcrtexact (PHP 4 >= 4.1.0)

Creates an exact but empty copy of a relation including indices

resource **dbplus_rcrtexact** (string name, resource relation, boolean overwrite) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rcrtexact() will create an exact but empty copy of the given `relation` under a new `name`. An existing relation by the same `name` will only be overwritten if `overwrite` is TRUE and no other process is currently using the relation.

dbplus_rcrtlike (PHP 4 >= 4.1.0)

Creates an empty copy of a relation with default indices

resource **dbplus_rcrtlike** (string name, resource relation, int flag) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rcrtexact() will create an empty copy of the given `relation` under a new `name`, but with default indices. An existing relation by the same `name` will only be overwritten if `overwrite` is TRUE and no other process is currently using the relation.

dbplus_resolve (PHP 4 >= 4.1.0)

Resolve host information for relation

int **dbplus_resolve** (string relation_name) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_resolve() will try to resolve the given *relation_name* and find out internal server id, real hostname and the database path on this host. The function will return an array containing these values under the keys 'sid', 'host' and 'host_path' or FALSE on error.

See also dbplus_tcl().

dbplus_restorepos (PHP 4 >= 4.1.0)

???

int **dbplus_restorepos** (resource relation, array tuple) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Not implemented yet.

dbplus_rkeys (PHP 4 >= 4.1.0)

Specify new primary key for a relation

resource **dbplus_rkeys** (resource relation, mixed domlist) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rkeys() will replace the current primary key for *relation* with the combination of domains specified by *domlist*.

domlist may be passed as a single domain name string or as an array of domain names.

dbplus_ropen (PHP 4 >= 4.1.0)

Open relation file local

resource **dbplus_ropen** (string name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_ropen() will open the relation *`file`* locally for quick access without any client/server overhead. Access is read only and only **dbplus_current()** and dbplus_next() may be applied to the returned relation.

dbplus_rquery (PHP 4 >= 4.1.0)

Perform local (raw) AQL query

int **dbplus_rquery** (string query, string dbpath) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rquery() performs a local (raw) AQL query using an AQL interpreter embedded into the db++ client library. **dbplus_rquery()** is faster than dbplus_aql() but will work on local data only.

dbplus_rrename (PHP 4 >= 4.1.0)

Rename a relation

int **dbplus_rrename** (resource relation, string name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rrename() will change the name of *`relation`* to *`name`*.

dbplus_rsecindex (PHP 4 >= 4.1.0)

Create a new secondary index for a relation

resource **dbplus_rsecindex** (resource relation, mixed domlist, int type) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rsecindex() will create a new secondary index for *`relation`* with consists of the domains specified by *`domlist`* and is of type *`type`*

`domlist` may be passed as a single domain name string or as an array of domain names.

dbplus_runlink (PHP 4 >= 4.1.0)

Remove relation from filesystem

int **dbplus_runlink** (resource relation) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_unlink() will close and remove the *`relation`*.

dbplus_rzap (PHP 4 >= 4.1.0)

Remove all tuples from relation

int **dbplus_rzap** (resource relation) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_rzap() will remove all tuples from *`relation`*.

dbplus_savepos (PHP 4 >= 4.1.0)

???

int **dbplus_savepos** (resource relation) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Not implemented yet.

dbplus_setindex (PHP 4 >= 4.1.0)

???

int **dbplus_setindex** (resource relation, string idx_name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Not implemented yet.

dbplus_setindexbynumber (PHP 4 >= 4.1.0)

???

int **dbplus_setindexbynumber** (resource relation, int idx_number) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Not implemented yet.

dbplus_sql (PHP 4 >= 4.1.0)

Perform SQL query

resource **dbplus_sql** (string query, string server, string dbpath) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Not implemented yet.

dbplus_tcl (PHP 4 >= 4.1.0)

Execute TCL code on server side

int **dbplus_tcl** (int sid, string script) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

A db++ server will prepare a TCL interpreter for each client connection. This interpreter will enable the server to execute TCL code provided by the client as a sort of stored procedures to improve the performance of database operations by avoiding client/server data transfers and context switches.

dbplus_tcl() needs to pass the client connection id the TCL `script` code should be executed by. dbplus_resolve() will provide this connection id. The function will return whatever the TCL code returns or a TCL error message if the TCL code fails.

See also dbplus_resolve().

dbplus_tremove (PHP 4 >= 4.1.0)

Remove tuple and return new current tuple

int **dbplus_tremove** (resource relation, array tuple [, array current]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_tremove() removes `tuple` from `relation` if it perfectly matches a tuple within the relation. `current`, if given, will contain the data of the new current tuple after calling **dbplus_tremove()**.

dbplus_undo (PHP 4 >= 4.1.0)

???

int **dbplus_undo** (resource relation) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Not implemented yet.

dbplus_undoprepare (PHP 4 >= 4.1.0)

???

int **dbplus_undoprepare** (resource relation) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Not implemented yet.

dbplus_unlockrel (PHP 4 >= 4.1.0)

Give up write lock on relation

int **dbplus_unlockrel** (resource relation) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_unlockrel() will release a write lock previously obtained by dbplus_lockrel().

dbplus_unselect (PHP 4 >= 4.1.0)

Remove a constraint from relation

int **dbplus_unselect** (resource relation) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Calling **dbplus_unselect()** will remove a constraint previously set by dbplus_find() on *relation*.

dbplus_update (PHP 4 >= 4.1.0)

Update specified tuple in relation

int **dbplus_update** (resource relation, array old, array new) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

dbplus_update() replaces the tuple given by *old* with the data from *new* if and only if *old* completely matches a tuple within *relation*.

dbplus_xlockrel (PHP 4 >= 4.1.0)

Request exclusive lock on relation

int **dbplus_xlockrel** (resource relation) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

dbplus_xlockrel() will request an exclusive lock on `relation` preventing even read access from other clients.

See also dbplus_xunlockrel().

dbplus_xunlockrel (PHP 4 >= 4.1.0)

Free exclusive lock on relation

int **dbplus_xunlockrel** (resource relation) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

dbplus_xunlockrel() will release an exclusive lock on `relation` previously obtained by dbplus_xlockrel().

XXIII. Direct IO functions

Introduzione

PHP supports the direct io functions as described in the Posix Standard (Section 6) for performing I/O functions at a lower level than the C-Language stream I/O functions (fopen, fread,..).

Requisiti

Queste funzioni sono disponibili nei moduli standard, che sono sempre disponibili.

Installation

To get these functions to work, you have to configure PHP with `--enable-dio`.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

One resource type is defined by this extension: a file descriptor returnded by dio_open().

Costanti Predefinite

Questa estensione non definisce alcuna costante.

dio_close (PHP 4 >= 4.2.0)

Closes the file descriptor given by fd

void **dio_close** (resource fd) \linebreak

The function **dio_close()** closes the file descriptor *resource*.

dio_fcntl (PHP 4 >= 4.2.0)

Performs a c library fcntl on fd

mixed **dio_fcntl** (resource fd, int cmd [, mixed arg]) \linebreak

The **dio_fcntl()** function performs the operation specified by *cmd* on the file descriptor *fd*. Some commands require additional arguments *args* to be supplied.

arg is an associative array, when *cmd* is F_SETLK or F_SETLLW, with the following keys:

- "start" - offset where lock begins
- "length" - size of locked area. zero means to end of file
- "wenth" - Where l_start is relative to: can be SEEK_SET, SEEK_END and SEEK_CUR
- "type" - type of lock: can be F_RDLCK (read lock), F_WRLCK (write lock) or F_UNLCK (unlock)

cmd can be one of the following operations:

- F_SETLK - Lock is set or cleared. If the lock is held by someone else **dio_fcntl()** returns -1.
- F_SETLKW - like F_SETLK, but in case the lock is held by someone else, **dio_fcntl()** waits until the lock is released.
- F_GETLK - **dio_fcntl()** returns an associative array (as described above) if someone else prevents lock. If there is no obstruction key "type" will set to F_UNLCK.
- F_DUPFD - finds the lowest numbered available file descriptor greater or equal than *arg* and returns them.

dio_open (PHP 4 >= 4.2.0)

Opens a new filename with specified permissions of flags and creation permissions of mode

resource **dio_open** (string filename, int flags [, int mode]) \linebreak

dio_open() opens a file and returns a new file descriptor for it, or -1 if any error occurred. If *flags* is O_CREAT, optional third parameter *mode* will set the mode of the file (creation permissions). The *flags* parameter can be one of the following options:

- O_RDONLY - opens the file for read access
- O_WRONLY - opens the file for write access
- O_RDWR - opens the file for both reading and writing

The *flags* parameter can also include any combination of the following flags:

- O_CREAT - creates the file, if it doesn't already exist
- O_EXCL - if both, O_CREAT and O_EXCL are set, **dio_open()** fails, if file already exists
- O_TRUNC - if file exists, and its opened for write access, file will be truncated to zero length.
- O_APPEND - write operations write data at the end of file
- O_NONBLOCK - sets non blocking mode

dio_read (PHP 4 >= 4.2.0)

Reads n bytes from fd and returns them, if n is not specified, reads 1k block

string **dio_read** (resource fd [, int n]) \linebreak

The function **dio_read()** reads and returns *n* bytes from file with descriptor *resource*. If *n* is not specified, **dio_read()** reads 1K sized block and returns them.

dio_seek (PHP 4 >= 4.2.0)

Seeks to pos on fd from whence

int **dio_seek** (resource fd, int pos, int whence) \linebreak

The function **dio_seek()** is used to change the file position of the file with descriptor *resource*. The parameter *whence* specifies how the position *pos* should be interpreted:

- SEEK_SET - specifies that *pos* is specified from the beginning of the file
- SEEK_CUR - Specifies that *pos* is a count of characters from the current file position. This count may be positive or negative
- SEEK_END - Specifies that *pos* is a count of characters from the end of the file. A negative count specifies a position within the current extent of the file; a positive count specifies a position past the current end. If you set the position past the current end, and actually write data, you will extend the file with zeros up to that position

dio_stat (PHP 4 >= 4.2.0)

Gets stat information about the file descriptor fd

array **dio_stat** (resource fd) \linebreak

Function **dio_stat()** returns information about the file with file descriptor *fd*. **dio_stat()** returns an associative array with the following keys:

- "device" - device
- "inode" - inode
- "mode" - mode
- "nlink" - number of hard links
- "uid" - user id
- "gid" - group id
- "device_type" - device type (if inode device)
- "size" - total size in bytes
- "blocksize" - blocksize
- "blocks" - number of blocks allocated
- "atime" - time of last access
- "mtime" - time of last modification
- "ctime" - time of last change

On error **dio_stat()** returns NULL.

dio_truncate (PHP 4 >= 4.2.0)

Truncates file descriptor fd to offset bytes

bool **dio_truncate** (resource fd, int offset) \linebreak

Function **dio_truncate()** causes the file referenced by *fd* to be truncated to at most *offset* bytes in size. If the file previously was larger than this size, the extra data is lost. If the file previously was shorter, it is unspecified whether the file is left unchanged or is extended. In the latter case the extended part reads as zero bytes. Returns 0 on success, otherwise -1.

dio_write (PHP 4 >= 4.2.0)

Writes data to fd with optional truncation at length

int **dio_write** (resource fd, string data [, int len]) \linebreak

The function **dio_write()** writes up to *len* bytes from *data* to file *fd*. If *len* is not specified, **dio_write()** writes all *data* to the specified file. **dio_write()** returns the number of bytes written to *fd*.

XXIV. Funzioni per le directory

Per funzioni correlate, quali dirname(), is_dir(), mkdir() e rmdir(), vedere la sezione Filesystem.

chdir (PHP 3, PHP 4 >= 4.0.0)

cambia directory

bool **chdir** (string directory) \linebreak

Cambia la directory in uso da parte del PHP in *directory*. Restituisce FALSE se non in grado di cambiare directory, TRUE altrimenti.

chroot (PHP 4 >= 4.0.5)

cambia la direcory di root

bool **chroot** (string directory) \linebreak

Cambia la directory di root del processo in esecuzione in *directory*. Restituisce FALSE se non in grado di cambiare la directory root, altrimenti TRUE.

> **Nota:** Non è intelligente usare questa funzione quando ci si trova nell'ambiente di lavoro di un webserver, poiché non è possibile resettare la directory root nuovamente a / alla fine della richiesta. Questa funzione si comporta correttamente solo quando eseguita come CGI.

dir (PHP 3, PHP 4 >= 4.0.0)

classe directory

```
 class dir {
  dir(string directory);
  string path;
  string read();
  void rewind();
  void close();
}
```

Un meccanismo pseudo orientato agli oggetti per leggere una directory. La *directory* data è aperta. Due proprietà sono disponibili nonappena la directory è stata aperta. La proprietà handle può essere usata in congiunzione ad altre funzioni relative alle directory, quali readdir(), rewinddir() e closedir(). La proprietà path è impostata alla directory che è stata aperta. Sono disponibili tre metodi: read (leggi), rewind (riavvolgi) e close (chiudi).

Fare attenzione al modo in cui il valore restituito dalla funzione **dir()** è controllato nell'esempio sotto riportato. Controlliamo esplicitamente che il valore restituito sia identico a (uguale a e dello stesso tipo di (fare riferimento a Comparison Operators per maggiori informazioni) FALSE altrimenti, ogni risultato il cui nome non venga valutato FALSE farà interrompere il ciclo.

Esempio 1. esempio dir()

```
$d = dir("/etc");
echo "Handle: ".$d->handle."<br>\n";
echo "Path: ".$d->path."<br>\n";
while ($entry = $d->read()) {
    echo $entry."<br>\n";
}
$d->close();
```

Nota: L'ordine nel quale vengono restituiti i dati dal metodo read è dipendente dal sistema usato.

Nota: Questo definisce la classe interna `Directory`, ciò significa che non sarà possibile definire una nuova classe con lo stesso nome. Per una lista completa delle classi predefinite presenti in PHP, fare riferimento a Predefined Classes.

closedir (PHP 3, PHP 4 >= 4.0.0)

chiude l'handle della directory

void **closedir** (resource dir_handle) \linebreak

Chiude il flusso directory indicato da *dir_handle*. Il flusso deve essere stato aperto in precedenza da opendir().

getcwd (PHP 4 >= 4.0.0)

restituisce la directory di lavoro in uso

string **getcwd** (void) \linebreak

Restituisce la directory di lavoro in uso al momento.

opendir (PHP 3, PHP 4 >= 4.0.0)

apre l'handle della directory

resource **opendir** (string percorso) \linebreak

Restituisce un handle della directory da usare nelle chiamate alle funzioni closedir(), readdir() e rewinddir().

Se `percorso` non è una directory valida o la directory non può essere aperta a causa di restrizioni sui permessi di accesso o a causa di errori del filesystem, **opendir()** restituisce `FALSE` e genera un errore PHP. Si può sopprimere l'output dell'errore di **opendir()** anteponendo '@' davanti al nome della funzione.

Esempio 1. esempio opendir()

```
<?php

if ($dir = @opendir("/tmp")) {
  while (($file = readdir($dir)) !== false) {
    echo "$file\n";
  }
  closedir($dir);
}

?>
```

Vedere anche is_dir().

readdir (PHP 3, PHP 4 >= 4.0.0)

legge una voce dall'handle della directory

string **readdir** (resource dir_handle) \linebreak

Restituisce il nomefile del file successivo della directory. I nomi dei file vengono restituiti secondo l'ordine in cui sono memorizzati nel filesystem.

Si faccia caso al modo in cui il valore restituito da **readdir()** viene controllato negli esempi successivi. Viene controllato esplicitamente che il valore restituito sia identico a (uguale a e dello stesso tipo di (vedere Comparison Operators per maggiori informazioni) `FALSE` altrimenti avverrebbe che ogni nome di directory il cui nome fosse valutato `FALSE` interromperebbe il loop.

Esempio 1. Elenca tutti i file presenti in una directory

```
// Nota che l'operatore !== non è esistito fino alla versione 4.0.0-RC2
<?php
if ($handle = opendir('/percorso/ai/file')) {
    echo "Handle della directory: $handle\n";
    echo "File:\n";

   /* Questa è la maniera corretta di eseguire un loop all'interno di una di-
rectory. */
   while (false !== ($file = readdir($handle))) {
```

```
        echo "$file\n";
    }

    /* Questa è la maniera SCORRETTA di eseguire un loop all'interno di una di-
rectory. */
    while ($file = readdir($handle)) {
        echo "$file\n";
    }

    closedir($handle);
}
?>
```

Nota che **readdir()** restituirà le voci `.` e `..`. Se non si vogliono ottenere queste, si possono semplicemente eliminare:

Esempio 2. Elenca tutti i file della directory in uso ed elimina `.` e `..`

```
<?php
if ($handle = opendir('.')) {
    while (false !== ($file = readdir($handle))) {
        if ($file != "." && $file != "..") {
            echo "$file\n";
        }
    }
    closedir($handle);
}
?>
```

Vedere anche is_dir().

rewinddir (PHP 3, PHP 4 >= 4.0.0)

riavvolge l'handle della directory

void **rewinddir** (resource dir_handle) \linebreak

Reimposta il flusso della directory indicato da *`dir_handle`* all'inizio della directory.

XXV. DOM XML functions

Introduzione

Attenzione

Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

The domxml extension has been overhauled in PHP Version 4.3.0 in favour of a better compliance of the DOM standard. The extension still contains many old functions, but they should not be used anymore. Especially those non object oriented functions should be avoided.

The extension allows you to operate on an XML document with the DOM API. It also provides a function domxml_xmltree() to turn the complete XML document into a tree of PHP objects. Currently this tree should be considered read-only - you can modify it but this would not make any sense since DomDocument_dump_mem() cannot be applied to it. Therefore, if you want to read an XML file and write a modified version use the DomDocument_create_element(), DomDocument_create_text_node(), **set_attribute()**, etc. and finally DomDocument_dump_mem() functions.

Requisiti

This extension make use of the GNOME xml library (http://www.xmlsoft.org). Download and install this library. You will need at least libxml-2.2.7.

Istallazione

This extension is only available if PHP was configured with `--with-dom=[DIR]`.

Deprecated functions

There a quite some functions which do not fit into the DOM standard and should not be used anymore as listed in the following table. The function DomNode_append_child() has changed its behaviour. It now actually adds a child and not a sibling. If this breaks your application use the non DOM function DomNode_append_sibling().

Tabella 1. Deprecated functions and its replacements

Old function	New function
xmldoc	domxml_open_mem()
xmldocfiel	domxml_open_file()

Old function	New function
domxml_new_xmldoc	domxml_new_doc()
domxml_dump_mem	DomDocument_dump_mem()
domxml_dump_mem_file	DomDocument_dump_file()
DomDocument_dump_mem_file	DomDocument_dump_file()
DomDocument_add_root	DomDocument_create_element() followed by DomNode_append_child()
DomDocument_dtd	DomDocument_doctype()
DomDocument_root	DomDocument_document_element()
DomDocument_children	DomNode_child_nodes()
DomDocument_imported_node	No replacement.
DomNode_add_child	Create a new node with e.g. DomDocument_create_element() und add it with DomNode_append_child().
DomNode_children	DomNode_child_nodes()
DomNode_parent	DomNode_parent_node()
DomNode_new_child	Create a new node with e.g. DomDocument_create_element() und add it with DomNode_append_child().
DomNode_set_content	Create a new node with e.g. DomDocument_create_text_node() und add it with DomNode_append_child().
DomNode_get_content	Content is just a text node and can be accessed with DomNode_child_nodes().
DomNode_set_content	Content is just a text node and can be added with DomNode_append_child().

Costanti Predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata compilata nel PHP o se è stata caricata dinamicamente a runtime.

Tabella 2. XML constants

Constant	Value	Description
`XML_ELEMENT_NODE` (integer)	1	Node is an element
`XML_ATTRIBUTE_NODE` (integer)	2	Node is an attribute
`XML_TEXT_NODE` (integer)	3	Node is a piece of text
`XML_CDATA_SECTION_NODE` (integer)	4	
`XML_ENTITY_REF_NODE` (integer)	5	

Constant	**Value**	**Description**
XML_ENTITY_NODE (integer)	6	Node is an entity like
XML_PI_NODE (integer)	7	Node is a processing instruction
XML_COMMENT_NODE (integer)	8	Node is a comment
XML_DOCUMENT_NODE (integer)	9	Node is a document
XML_DOCUMENT_TYPE_NODE (integer)	10	
XML_DOCUMENT_FRAG_NODE (integer)	11	
XML_NOTATION_NODE (integer)	12	
XML_GLOBAL_NAMESPACE (integer)	1	
XML_LOCAL_NAMESPACE (integer)	2	
XML_HTML_DOCUMENT_NODE (integer)		
XML_DTD_NODE (integer)		
XML_ELEMENT_DECL_NODE (integer)		
XML_ATTRIBUTE_DECL_NODE (integer)		
XML_ENTITY_DECL_NODE (integer)		
XML_NAMESPACE_DECL_NODE (integer)		
XML_ATTRIBUTE_CDATA (integer)		
XML_ATTRIBUTE_ID (integer)		
XML_ATTRIBUTE_IDREF (integer)		
XML_ATTRIBUTE_IDREFS (integer)		
XML_ATTRIBUTE_ENTITY (integer)		
XML_ATTRIBUTE_NMTOKEN (integer)		
XML_ATTRIBUTE_NMTOKENS (integer)		
XML_ATTRIBUTE_ENUMERATION (integer)		
XML_ATTRIBUTE_NOTATION (integer)		
XPATH_UNDEFINED (integer)		

Constant	Value	Description
XPATH_NODESET (integer)		
XPATH_BOOLEAN (integer)		
XPATH_NUMBER (integer)		
XPATH_STRING (integer)		
XPATH_POINT (integer)		
XPATH_RANGE (integer)		
XPATH_LOCATIONSET (integer)		
XPATH_USERS (integer)		
XPATH_NUMBER (integer)		

Classes

The API of the module follows the DOM Level 2 standard as close as possible. Consequently the API is fully object oriented. It is a good idea to have the DOM standard available when using this module. Though the API is object oriented there are many functions which can be called in a non-object oriented way by passing the object to operate on as the first argument. These function are mainly to retain compatibilty to older versions of the extension but are not encouraged to use anymore in new developments.

This API differs from the official DOM API in two points. First, all class attributes are implemented as functions with the same name and secondly the function names follow the PHP naming convention. This means that a DOM function lastChild() will be written as last_child().

This module defines a number of classes, which are listed — including their method — in the following tables. Classes with an equivalent in the DOM Standard are named DOMxxx.

Tabella 3. List of classes

Class name	Parent classes
DomAttribute	DomNode
DomCData	DomNode
DomComment	DomCData : DomNode
DomDocument	DomNode
DomDocumentType	DomNode
DomElement	DomNode
DomEntity	DomNode
DomEntityReference	DomNode
DomProcessingInstruction	DomNode
DomText	DomCData : DomNode
Parser	Currently still called DomParser
XPathContext	

Tabella 4. DomDocument class (DomDocument : DomNode)

Method name	Function name	Remark
doctype	DomDocument_doctype()	
document_elemnent	DomDocument_document_element()	
create_element	DomDocument_create_element()	
create_text_node	DomDocument_create_text_node()	
create_comment	DomDocument_create_comment()	
create_cdata_section	DomDocument_create_cdata_section()	
create_processing_instruction	DomDocument_create_processing_instruction()	
create_attribute	DomDocument_create_attribute()	
create_entity_reference	DomDocument_create_entity_reference()	
get_elements_by_tagname	DomDocument_get_elements_by_tagname()	
get_element_by_id	DomDocument_get_element_by_id()	
dump_mem	DomDocument_dump_mem()	not DOM standard
dump_file	DomDocument_dump_file()	not DOM standard
html_dump_mem	DomDocument_html_dump_mem()	not DOM standard
xpath_init	xpath_init	not DOM standard
xpath_new_context	xpath_new_context	not DOM standard
xptr_new_context	xptr_new_context	not DOM standard

Tabella 5. DomElement class (DomElement : DomNode)

Method name	Function name	Remark
tagname	DomElement_tagname()	
get_attribute	DomElement_get_attribute()	
set_attribute	DomElement_set_attribute()	
remove_attribute	DomElement_remove_attribute()	
get_attribute_node	DomElement_get_attribute_node()	

Method name	Function name	Remark
set_attribute_node	DomElement_set_attribute_node()	
get_elements_by_tagname	DomElement_get_elements_by_tagname()	
has_attribute	DomElement_has_attribute()	

Tabella 6. DomNode class

Method name	Remark
DomNode_node_name()	
DomNode_node_value()	
DomNode_node_type()	
DomNode_last_child()	
DomNode_first_child()	
DomNode_child_nodes()	
DomNode_previous_sibling()	
DomNode_next_sibling()	
DomNode_parent_node()	
DomNode_owner_document()	
DomNode_insert_before()	
DomNode_append_child()	
DomNode_append_sibling()	Not in DOM standard. This function emulates the former behaviour of DomNode_append_child().
DomNode_remove_child()	
DomNode_has_child_nodes()	
DomNode_has_attributes()	
DomNode_clone_node()	
DomNode_attributes()	
DomNode_unlink_node()	Not in DOM standard
DomNode_replace_node()	Not in DOM standard
DomNode_set_content()	Not in DOM standard, deprecated
DomNode_get_content()	Not in DOM standard, deprecated
DomNode_dump_node()	Not in DOM standard
DomNode_is_blank_node()	Not in DOM standard

Tabella 7. DomAttribute class (DomAttribute : DomNode)

Method name		Remark
name	DomAttribute_name()	
value	DomAttribute_value()	

Method name		Remark
specified	DomAttribute_specified()	

Tabella 8. DomProcessingInstruction class (DomProcessingInstruction : DomNode)

Method name	Function name	Remark
target	DomProcessingInstruction_target()	
data	DomProcessingInstruction_data()	

Tabella 9. Parser class

Method name	Function name	Remark
add_chunk	**Parser_add_chunk()**	
end	**Parser_end()**	

Tabella 10. XPathContext class

Method name	Function name	Remark
eval	**XPathContext_eval()**	
eval_expression	**XPathContext_eval_expression()**	
register_ns	**XPathContext_register_ns()**	

Tabella 11. DomDocumentType class (DomDocumentType : DomNode)

Method name	Function name	Remark
name	DomDocumentType_name()	
entities	DomDocumentType_entities()	
notations	DomDocumentType_notations()	
public_id	DomDocumentType_public_id()	
system_id	DomDocumentType_system_id()	
internal_subset	DomDocumentType_internal_subset()	

The classes DomDtd is derived from DomNode. DomComment is derived from DomCData

Esempi

Many examples in this reference require a xml string. Instead of repeating this string in any example it will be put into a file and be included by each example. This include file is shown in the following example section. You could also create an xml document read it with **DomDocument_open_file()**.

Esempio 1. Include file example.inc with xml string

```
<?php
$xmlstr = "<?xml version='1.0' standalone='yes'?>
<!DOCTYPE chapter SYSTEM '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'
[ <!ENTITY sp \"spanish\">
]>
<!-- lsfj  -->
<chapter language='en'><title language='en'>Title</title>
 <para language='ge'>
  &amp;sp;
  <!-- comment -->
  <informaltable ID='findme' language='&amp;sp;'>
   <tgroup cols='3'>
    <tbody>
     <row><entry>a1</entry><entry
morerows='1'>b1</entry><entry>c1</entry></row>
<row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
</chapter>";
?>
```

DomAttribute->name (unknown)

Returns name of attribute

bool **DomAttribute->name** (void) \linebreak

This function returns the name of the attribute.

See also DomAttribute_value().

DomAttribute->specified (unknown)

Checks if attribute is specified

bool **DomAttribute->specified** (void) \linebreak

Check DOM standard for a detailed explanation.

DomAttribute->value (unknown)

Returns value of attribute

bool **DomAttribute->value** (void) \linebreak

This function returns the value of the attribute.

See also DomAttribute_name().

DomDocument->add_root [deprecated] (unknown)

Adds a root node

resource **DomDocument->add_root** (string name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Adds a root element node to a dom document and returns the new node. The element name is given in the passed parameter.

Esempio 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->add_root("HTML");
$head = $root->new_child("HEAD", "");
$head->new_child("TITLE", "Hier der Titel");
echo htmlentities($doc->dump_mem());
?>
```

DomDocument->create_attribute (unknown)

Create new attribute

object **DomDocument->create_attribute** (string name, string value) \linebreak

This function returns a new instance of class `DomAttribute`. The name of the attribute is the value of the first parameter. The value of the attribute is the value of the second parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_cdata_section (unknown)

Create new cdata node

string **DomDocument->create_cdata_section** (string content) \linebreak

This function returns a new instance of class `DomCData`. The content of the cdata is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_comment (unknown)

Create new comment node

object **DomDocument->create_comment** (string content) \linebreak

This function returns a new instance of class `DomComment`. The content of the comment is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_element (unknown)

Create new element node

object **DomDocument->create_element** (string name) \linebreak

This function returns a new instance of class `DomElement`. The tag name of the element is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), **DomDocument_create_text()**, DomDocument_create_comment(), DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_entity_reference (unknown)

object **DomDocument->create_entity_reference** (string content) \linebreak

This function returns a new instance of class `DomEntityReference`. The content of the entity reference is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_processing_instruction(), DomDocument_create_attribute(), DomNode_insert_before().

DomDocument->create_processing_instruction (unknown)

Creates new PI node

string **DomDocument->create_processing_instruction** (string content) \linebreak

This function returns a new instance of class `DomCData`. The content of the pi is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), **DomDocument_create_text()**, DomDocument_create_cdata_section(), DomDocument_create_attribute(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->create_text_node (unknown)

Create new text node

object **DomDocument->create_text_node** (string content) \linebreak

This function returns a new instance of class `DomText`. The content of the text is the value of the passed parameter. This node will not show up in the document unless it is inserted with e.g. DomNode_append_child().

The return value is false if an error occured.

See also DomNode_append_child(), DomDocument_create_element(), DomDocument_create_comment(), **DomDocument_create_text()**, DomDocument_create_attribute(), DomDocument_create_processing_instruction(), DomDocument_create_entity_reference(), DomNode_insert_before().

DomDocument->doctype (unknown)

Returns the document type

object **DomDocument->doctype** (void) \linebreak

This function returns an object of class `DomDocumentType`. In versions of PHP before 4.3 this has been the class `Dtd`, but the DOM Standard does not know such a class.

See also the methods of class `DomDocumentType`.

DomDocument->document_element (unknown)

Returns root element node

object **DomDocument->document_element** (void) \linebreak

This function returns the root element node of a document.

The following example returns just the element with name CHAPTER and prints it. The other node -- the comment -- is not returned.

Esempio 1. Retrieving root element

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
print_r($root);
?>
```

DomDocument->dump_file (unknown)

Dumps the internal XML tree back into a file

string **DomDocument->dump_file** (string filename [, bool compressionmode [, bool format]]) \linebreak

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The *format* specifies whether the output should be neatly formatted, or not. The first parameter specifies the name of the filename and the second parameter, whether it should be compressed or not.

Esempio 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
$doc->dump_file("/tmp/test.xml", false, true);
?>
```

See also DomDocument_dump_mem() DomDocument_html_dump_mem().

DomDocument->dump_mem (unknown)

Dumps the internal XML tree back into a string

string **DomDocument->dump_mem** ([bool format]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Creates an XML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below. The `format` specifies whether the output should be neatly formatted, or not.

Esempio 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->dump_mem(true));
echo "</PRE>";
?>
```

Nota: The first parameter was added in PHP 4.3.0.

See also DomDocument_dump_file(), DomDocument_html_dump_mem().

DomDocument->get_element_by_id (unknown)

Searches for an element with a certain id

object **DomDocument->get_element_by_id** (string id) \linebreak

This function is similar to DomDocument_get_elements_by_tagname() but searches for an element with a given id. According to the DOM standard this requires a DTD which defines the attribute ID to be of type ID, though the current implementation simply does an xpath search for "//*[@ID = ’%s’]". This does not comply to the DOM standard which requires to return null if it is not known which attribute is of type id. This behaviour is likely to be fixed, so do not rely on the current behaviour.

See also DomDocument_get_elements_by_tagname()

DomDocument->get_elements_by_tagname (unknown)

array **DomDocument->get_elements_by_tagname** (string name) \linebreak

See also DomDocument_add_root()

DomDocument->html_dump_mem (unknown)

Dumps the internal XML tree back into a string as HTML

string **DomDocument->html_dump_mem** (void) \linebreak

Creates an HTML document from the dom representation. This function usually is called after building a new dom document from scratch as in the example below.

Esempio 1. Creating a simple HTML document header

```
<?php
$doc = domxml_new_doc("1.0");
$root = $doc->create_element("HTML");
$root = $doc->append_child($root);
$head = $doc->create_element("HEAD");
$head = $root->append_child($head);
$title = $doc->create_element("TITLE");
$title = $head->append_child($title);
$text = $doc->create_text_node("This is the title");
$text = $title->append_child($text);
echo "<PRE>";
echo htmlentities($doc->html_dump_mem());
echo "</PRE>";
?>
```

See also DomDocument_dump_file(), DomDocument_html_dump_mem().

DomDocumentType->entities (unknown)

Returns list of entities

array **DomDocumentType->entities** (void) \linebreak

> **Attenzione**
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomDocumentType->internal_subset (unknown)

Returns internal subset

bool **DomDocumentType->internal_subset** (void) \linebreak

> **Attenzione**
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomDocumentType->name (unknown)

Returns name of document type

string **DomDocumentType->name** (void) \linebreak

This function returns the name of the document type.

DomDocumentType->notations (unknown)

Returns list of notations

array **DomDocumentType->notations** (void) \linebreak

> **Attenzione**
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomDocumentType->public_id (unknown)

Returns public id of document type

string **DomDocumentType->public_id** (void) \linebreak

This function returns the public id of the document type.

The following example echos nothing.

Esempio 1. Retrieving the public id

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$doctype = $dom->doctype();
echo $doctype->public_id();
?>
```

DomDocumentType->system_id (unknown)

Returns system id of document type

string **DomDocumentType->system_id** (void) \linebreak

Returns the system id of the document type.

The following example echos '/share/sgml/Norman_Walsh/db3xml10/db3xml10.dtd'.

Esempio 1. Retrieving the system id

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$doctype = $dom->doctype();
```

```
echo $doctype->system_id();
?>
```

DomElement->get_attribute (unknown)

Returns value of attribute

object **DomElement->get_attribute** (string name) \linebreak

Returns the attribute with name `name` of the current node.

See also DomElement_set_attribute()

DomElement->get_attribute_node (unknown)

Returns value of attribute

object **DomElement->get_attribute_node** (object attr) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomElement->get_elements_by_tagname (unknown)

Adds new attribute

bool **DomElement->get_elements_by_tagname** (string name) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomElement->has_attribute (unknown)

Adds new attribute

bool **DomElement->has_attribute** (string name) \linebreak

> **Attenzione**
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomElement->remove_attribute (unknown)

Adds new attribute

bool **DomElement->remove_attribute** (string name) \linebreak

> **Attenzione**
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomElement->set_attribute (unknown)

Adds new attribute

bool **DomElement->set_attribute** (string name, string value) \linebreak

Sets an attribute with name *name* ot the given value. If the attribute does not exist, it will be created.

Esempio 1. Setting an attribute

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

See also DomElement_get_attribute()

DomElement->set_attribute_node (unknown)

Adds new attribute

bool **DomElement->set_attribute_node** (object attr) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomElement->tagname (unknown)

Returns name of element

string **DomElement->tagname** (void) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomNode->append_child (unknown)

Adds new child at the end of the children

object **DomNode->append_child** (object newnode) \linebreak

This functions appends a child to an existing list of children or creates a new list of children. The child can be created with e.g. DomDocument_create_element(), **DomDocument_create_text()** etc. or simply by using any other node.

Before a new child is appended it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of a xml document. The return value is the appended child. If you plan to do further modifications on the appended child you must use the returned node.

The following example will add a new element node to a fresh document and sets the attribute "align" to "left".

Esempio 1. Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$newnode = $doc->append_child($node);
$newnode->set_attribute("align", "left");
?>
```

The above example could also be written as the following:

Esempio 2. Adding a child

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node->set_attribute("align", "left");
$newnode = $doc->append_child($node);
?>
```

A more comples example is the one below. It first searches for a certain element, duplicates it including its children and adds it as a sibling. Finally a new attribute is added to one of the children of the new sibling and the whole document is dumped.

Esempio 3. Adding a child

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$parent = $element->parent_node();
$newnode = $parent->append_child($element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<PRE>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

The above example could also be done with DomNode_insert_before() instead of DomNode_append_child().

See also DomNode_insert_before().

DomNode->append_sibling (unknown)

Adds new sibling to a node

object **DomNode->append_sibling** (object newnode) \linebreak

This functions appends a sibling to an existing node. The child can be created with e.g. DomDocument_create_element(), **DomDocument_create_text()** etc. or simply by using any other node.

Before a new sibling is added it is first duplicated. Therefore the new child is a completely new copy which can be modified without changing the node which was passed to this function. If the node passed has children itself, they will be duplicated as well, which makes it quite easy to duplicate large parts of a xml document. The return value is the added sibling. If you plan to do further modifications on the added sibling you must use the returned node.

This function has been added to provide the behaviour of DomNode_append_child() as it works till PHP 4.2.

See also **DomNode_append_before()**.

DomNode->attributes (unknown)

Returns list of attributes

array **DomNode->attributes** (void) \linebreak

This function only returns an array of attributes if the node is of type XML_ELEMENT_NODE.

DomNode->child-nodes (unknown)

Returns children of node

array **DomNode->child_nodes** (void) \linebreak

Returns all children of the node.

See also DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->clone_node (unknown)

Clones a node

object **DomNode->clone_node** (void) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomNode->dump_node (unknown)

Dumps a single node

string **DomNode->dump_node** (void) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

See also DomDocument_dump_mem().

DomNode->first_child (unknown)

Returns first child of node

bool **DomNode->first_child** (void) \linebreak

Returns the first child of the node.

See also DomNode_last_child(), DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->get_content (unknown)

Gets content of node

string **DomNode->get_content** (void) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomNode->has_attributess (unknown)

Checks if node has attributes

bool **DomNode->has_attributes** (void) \linebreak

This function checks if the node has attributes.

See also DomNode_has_child_nodes().

DomNode->has_child_nodes (unknown)

Checks if node has children

bool **DomNode->has_child_nodes** (void) \linebreak

This function checks if the node has children.

See also DomNode_child_nodes().

DomNode->insert_before (unknown)

Inserts new node as child

object **DomNode->insert_before** (object newnode, object refnode) \linebreak

This function inserts the new node *newnode* right before the node *refnode*. The return value is the inserted node. If you plan to do further modifications on the appended child you must use the returned node.

DomNode_insert_before() is very similar to DomNode_append_child() as the following example shows which does the same as the example at DomNode_append_child().

Esempio 1. Adding a child

```
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}
```

```
$elements = $dom->get_elements_by_tagname("informaltable");
print_r($elements);
$element = $elements[0];

$newnode = $element->insert_before($element, $element);
$children = $newnode->children();
$attr = $children[1]->set_attribute("align", "left");

echo "<PRE>";
$xmlfile = $dom->dump_mem();
echo htmlentities($xmlfile);
echo "</PRE>";
```

See also DomNode_append_child().

DomNode->is_blank_node (unknown)

Checks if node is blank

bool **DomNode->is_blank_node** (void) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomNode->last_child (unknown)

Returns last child of node

object **DomNode->last_child** (void) \linebreak

Returns the last child of the node.

See also DomNode_first_child(), DomNode_next_sibling(), DomNode_previous_sibling().

DomNode->next_sibling (unknown)

Returns the next sibling of node

object **DomNode->next_sibling** (void) \linebreak

This function returns the next sibling of the current node. If there is no next sibling it returns false. You can use this function to iterate over all children of a node as shown in the example.

Esempio 1. Iterate over children

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$child = $element->first_child();

while($child) {
   print_r($child);
   $child = $child->next_sibling();
}
?>
```

See also DomNode_previous_sibling().

DomNode->node_name (unknown)

Returns name of node

string **DomNode->node_name** (void) \linebreak

Returns name of the node. The name has different meanings for the different types of nodes as illustrated in the following table.

Tabella 1. Meaning of value

Type	Meaning
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	#cdata-section
DomComment	#comment
DomDocument	#document
DomDocumentType	document type name
DomElement	tag name
DomEntity	name of entity
DomEntityReference	name of entity reference

Type	Meaning
DomNotation	notation name
DomProcessingInstruction	target
DomText	#text

DomNode->node_type (unknown)

Returns type of node

int **DomNode->node_type** (void) \linebreak

Returns the type of the node. All possible types are listed in the table in the introduction.

DomNode->node_value (unknown)

Returns value of a node

string **DomNode->node_value** (void) \linebreak

Returns value of the node. The value has different meanings for the different types of nodes as illustrated in the following table.

Tabella 1. Meaning of value

Type	Meaning
DomAttribute	value of attribute
DomAttribute	
DomCDataSection	content
DomComment	content of comment
DomDocument	null
DomDocumentType	null
DomElement	null
DomEntity	null
DomEntityReference	null
DomNotation	null
DomProcessingInstruction	entire content without target
DomText	content of text

DomNode->owner_document (unknown)

Returns the document this node belongs to

object **DomNode->owner_document** (void) \linebreak

This function returns the document the current node belongs to.

The following example will create two identical lists of children.

Esempio 1. Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->owner_document();
$children = $doc2->children();
print_r($children);
?>
```

See also DomNode_insert_before().

DomNode->parent_node (unknown)

Returns the parent of the node

object **DomNode->parent_node** (void) \linebreak

This function returns the parent node.

The following example will show two identical lists of children.

Esempio 1. Finding the document of a node

```
<?php
$doc = domxml_new_doc("1.0");
$node = $doc->create_element("para");
$node = $doc->append_child($node);
$children = $doc->children();
print_r($children);

$doc2 = $node->parent_node();
$children = $doc2->children();
print_r($children);
?>
```

DomNode->prefix (unknown)

Returns name space prefix of node

string **DomNode->prefix** (void) \linebreak

Returns the name space prefix of the node.

DomNode->previous_sibling (unknown)

Returns the previous sibling of node

object **DomNode->previous_sibling** (void) \linebreak

This function returns the previous sibling of the current node.

See also DomNode_next_sibling().

DomNode->remove_child (unknown)

Removes child from list of children

object **DomNode->remove_child** (object oldchild) \linebreak

This functions removes a child from a list of children. If child cannot be removed or is not a child the function will return false. If the child could be removed the functions returns the old child.

Esempio 1. Removing a child

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$elements = $dom->get_elements_by_tagname("tbody");
$element = $elements[0];
$children = $element->child_nodes();
$child = $element->remove_child($children[0]);

echo "<PRE>";
```

```
$xmlfile = $dom->dump_mem(true);
echo htmlentities($xmlfile);
echo "</PRE>";
?>
```

See also DomNode_append_child().

DomNode->replace_child (unknown)

Replaces a child

object **DomNode->replace_child** (object oldnode, object newnode) \linebreak

This function replaces the child *oldnode* with the passed new node. If the new node is already a child it will not be added a second time. If the old node cannot be found the function returns false. If the replacement succeds the old node is returned.

See also DomNode_append_child()

DomNode->replace_node (unknown)

Replaces node

object **DomNode->replace_node** (object newnode) \linebreak

This function replaces an existing node with the passed new node. Before the replacement *newnode* is copied if it has a parent to make sure a node which is already in the document will not be inserted a second time. This behaviour enforces doing all modifications on the node before the replacement or to refetch the inserted node afterwards with functions like DomNode_first_child(), DomNode_child_nodes() etc..

See also DomNode_append_child()

DomNode->set_content (unknown)

Sets content of node

bool **DomNode->set_content** (void) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomNode->set_name (unknown)

Sets name of node

bool **DomNode->set_name** (void) \linebreak

Sets name of node.

See also DomNode_node_name().

DomNode->unlink_node (unknown)

Deletes node

object **DomNode->unlink_node** (void) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomProcessingInstruction->data (unknown)

Returns data of pi node

string **DomProcessingInstruction->data** (void) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

DomProcessingInstruction->target (unknown)

Returns target of pi node

string **DomProcessingInstruction->target** (void) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

domxml_new_doc (unknown)

Creates new empty XML document

object **domxml_new_doc** (string version) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Creates a new dom document from scratch and returns it.

See also DomDocument_add_root()

domxml_open_file (unknown)

Creates a DOM object from XML file

object **domxml_open_file** (string filename) \linebreak

The function parses the XML document in the file named *filename* and returns an object of class "Dom document", having the properties as listed above. The file is accessed read-only.

Esempio 1. Opening a xml document from a file

```
<?php

if(!$dom = domxml_open_file("example.xml")) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
?>
```

See also domxml_open_mem(), domxml_new_doc().

domxml_open_mem (unknown)

Creates a DOM object of an XML document

object **domxml_open_mem** (string str) \linebreak

The function parses the XML document in *str* and returns an object of class "Dom document", having the properties as listed above. This function, domxml_open_file() or domxml_new_doc() must be called before any other function calls.

Esempio 1. Opening a xml document in a string

```
<?php
include("example.inc");

if(!$dom = domxml_open_mem($xmlstr)) {
  echo "Error while parsing the document\n";
  exit;
}

$root = $dom->document_element();
?>
```

See also domxml_open_file(), domxml_new_doc().

domxml_version (PHP 4 >= 4.1.0)

Get XML library version

string **domxml_version** (void) \linebreak

This function returns the version of the XML library version currently used.

domxml_xmltree (unknown)

Creates a tree of PHP objects from an XML document

object **domxml_xmltree** (string str) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

The function parses the XML document in *`str`* and returns a tree PHP objects as the parsed document. This function is isolated from the other functions, which means you cannot access the tree with any of the other functions. Modifying it, for example by adding nodes, makes no sense since there is currently no way to dump it as an XML file. However this function may be valuable if you want to read a file and investigate the content.

xpath_eval (PHP 4 >= 4.0.4)

Evaluates the XPath Location Path in the given string

array **xpath_eval** (object xpath context) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

See also xpath_new_context()

xpath_eval_expression (PHP 4 >= 4.0.4)

Evaluates the XPath Location Path in the given string

array **xpath_eval_expression** (object xpath_context) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

See also xpath_eval()

xpath_new_context (PHP 4 >= 4.0.4)

Creates new xpath context

object **xpath_new_context** (object dom document) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

See also xpath_eval()

xptr_eval (PHP 4 >= 4.0.4)

Evaluate the XPtr Location Path in the given string

int **xptr_eval** ([object xpath_context, string eval_str]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xptr_new_context (PHP 4 >= 4.0.4)

Create new XPath Context

string **xptr_new_context** ([object doc_handle]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

XXVI. Funzioni .NET

dotnet_load (unknown)

Carica un modulo DOTNET

int **dotnet_load** (string assembly_name [, string datatype_name [, int codepage]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

XXVII. Funzioni di gestione degli errori e di logging

Le seguenti sono le funzioni per la gestione degli errori ed il logging. Esse permettono di definire regole personalizzate per la gestione degli errori, e anche di modificarne la modalità della gestione stessa. Ciò permette di cambiare ed integrare i messaggi di errore adattandoli alle vostre esigenze.

Grazie alle funzioni di logging, è possibile inviare messaggi direttamente ad altre macchine, alla posta elettronica (o ad un gateway pager o di posta elettronica!), ai log di sistema, ecc., in modo da effettuare il log e controllare selettivamente le parti più importanti delle vostre applicazioni e siti web.

Le funzioni di restituzione dell'errore consentono la personalizzazione del livello e del tipo di errore, dal semplice avviso sino a funzioni personalizzate restituite durante gli errori.

error_log (PHP 3, PHP 4 >= 4.0.0)

invia un messaggio di errore

int **error_log** (string messaggio [, int tipo_messaggio [, string destinazione [, string header_extra]]]) \line-break

Invia un messaggio di errore la log del server web, ad una porta TCP o ad un file. Il primo parametro, `messaggio`, è il messaggio di errore che deve essere registrato. Il secondo parametro, `tipo_messaggio` indica la destinazione del messaggio:

Tabella 1. error_log() tipi di log

0	`messaggio` è inviato al log di sistema di PHP, utilizzando il sistema di log del Sistema Operativo o un file, a seconda di come sia impostata la direttiva di configurazione error_log.
1	`messaggio` è inviato via posta elettronica all'indirizzo indicato nel parametro `destinazione` parameter. Questo è l'unico tipo di messaggio nel quale viene usato il quarto parametro, `headers_extra`. Questo tipo di messaggio utilizza la stessa funzione interna di mail().
2	`messaggio` viene inviato attraverso la connessione di debug di PHP. Questa opzione è disponibile solo nel caso che il debug remoto sia stato abilitato. In questo caso, il parametro `destinazione` specifica il nome dell'host o l'indirizzo IP e opzionalmente, numero di porta, del socket che riceverà l'informazione di debug.
3	`messaggio` è aggiunto al file `destinazione`.

Attenzione

Il debug remoto via TCP/IP è una caratteristica di PHP 3 *non* disponibile in PHP 4.

Esempio 1. error_log() esempi

```
// Invia notifica via log del server se non è possibile
// connettersi al database.
if (!Ora_Logon ($username, $password)) {
    error_log ("Database Oracle non disponibile!", 0);
}

// Notifica via posta elettronica all'amministratore se esauriscono i FOO
if (!($foo = allocate_new_foo()) {
```

```
    error_log ("Problemi seri, FOO esauriti!", 1,
               "operator@mydomain.com");
}

// altri modi per chiamare error_log():
error_log ("Problema!", 2, "127.0.0.1:7000");
error_log ("Problema!", 2, "loghost");
error_log ("Problema!", 3, "/var/tmp/my-errors.log");
```

error_reporting (PHP 3, PHP 4 >= 4.0.0)

definisce quali errori di PHP vengono restituiti

int **error_reporting** ([int livello]) \linebreak

Definisce il livello di restituzione di errore di PHP e ritorna il vecchio livello. Il livello di restituzione dell'errore può essere una maschera di bit o una costante named. L'utilizzo delle costanti named è caldamente consigliato per assicurare la compatibilità con versioni future. All'aggiungere di livelli di errore, la gamma degli interi viene incrementata, perciò vecchi livelli di errore basati sull'intero non si comporteranno sempre come ci si aspetta.

Esempio 1. Error Integer changes

```
error_reporting (55);   // Equivalente a E_ALL ^ E_NOTICE in PHP 3

/* ...in PHP 4, '55' indica (E_ERROR | E_WARNING | E_PARSE |
E_CORE_ERROR | E_CORE_WARNING) */

error_reporting (2039); // Equivalente a E_ALL ^ E_NOTICE in PHP 4

error_reporting (E_ALL ^ E_NOTICE); // Il medesimo in PHP 3 e 4
```

Seguite i collegamenti delle costanti per conoscerne il significato:

Tabella 1. error_reporting() valori dei bit

valore	costante
1	E_ERROR
2	E_WARNING
4	E_PARSE
8	E_NOTICE
16	E_CORE_ERROR
32	E_CORE_WARNING
64	E_COMPILE_ERROR

valore	costante
128	E_COMPILE_WARNING
256	E_USER_ERROR
512	E_USER_WARNING
1024	E_USER_NOTICE
2047	E_ALL

Esempio 2. esempi error_reporting()

```
// Turn off all error reporting
error_reporting (0);

// Report simple running errors
error_reporting  (E_ERROR | E_WARNING | E_PARSE);

// Reporting E_NOTICE can be good too (to report uninitialized
// variables or catch variable name misspellings)
error_reporting (E_ERROR | E_WARNING | E_PARSE | E_NOTICE);

// Report all PHP errors (use bitwise  63  in PHP 3)
error_reporting (E_ALL);
```

restore_error_handler (PHP 4)

Ripristina la precedente funzione di gestione dell'errore

void **restore_error_handler** (void) \linebreak

Utilizzata dopo la modifica della funzione di gestione degli errori con set_error_handler(), per ripristinare la precedente modalità di gestione (che può essere quella di configurazione o una definita dall'utente).

Vedere anche error_reporting(), set_error_handler(), trigger_error(), user_error()

set_error_handler (PHP 4)

Configura una funzione di gestione dell'errore definita dall'utente.

string **set_error_handler** (string error_handler) \linebreak

Configura una funzione utente (*error_handler* per gestire gli errori in uno script. Restituisce, se esistente, il precedente gestore degli errori, o `FALSE` in caso di errore. Questa funzione può essere utilizzata per definire un sistema personalizzato di gestione degli errori durante l'esecuzione, per

esempio in applicazioni dove sia necessario svuotare dati o file in caso di un determinato errore critico, o quando sia necessario, in determinate condizioni, attivare un errore (con trigger_error())

La funzione utente richiede 2 parametri: il codice errore e una stringa descrittiva dell'errore. Da PHP 4.0.2, è possibile opzionalmente fornire altri 3 parametri aggiuntivi: il nome del file, il numero di riga e il contesto dove è avvenuto l'errore (un array che punto alla tabella dei simboli attiva nel punto in cui è avvenuto l'errore).

L'esempio sottostante mostra la gestione delle eccezioni interne attivando gli errori e gestendoli tramite una funzione definita dall'utente:

Esempio 1. Gestione errori con set_error_handler() e trigger_error()

```
<?php

// ridefinisce la costante dell'errore utente - solo PHP 4
define (FATAL,E_USER_ERROR);
define (ERROR,E_USER_WARNING);
define (WARNING,E_USER_NOTICE);

// configura il livello di restituzione errore per questo script
error_reporting (FATAL | ERROR | WARNING);

// funzione di gestione dell'errore
function myErrorHandler ($errno, $errstr, $errfile, $errline) {
  switch ($errno) {
  case FATAL:
    echo "<b>FATAL</b> [$errno] $errstr<br>\n";
    echo "  Fatal error in line ".$errline." of file ".$errfile;
    echo ", PHP ".PHP_VERSION." (".PHP_OS.")<br>\n";
    echo "Aborting...<br>\n";
    exit 1;
    break;
  case ERROR:
    echo "<b>ERROR</b> [$errno] $errstr<br>\n";
    break;
  case WARNING:
    echo "<b>WARNING</b> [$errno] $errstr<br>\n";
    break;
    default:
    echo "Unkown error type: [$errno] $errstr<br>\n";
    break;
  }
}

// funzione di prova del gestore di errore
function scale_by_log ($vect, $scale) {
  if ( !is_numeric($scale) || $scale <= 0 )
    trigger_error("log(x) for x <= 0 is undefined, you used: scale = $scale",
      FATAL);
  if (!is_array($vect)) {
    trigger_error("Incorrect input vector, array of values expected", ERROR);
    return null;
  }
  for ($i=0; $i<count($vect); $i++) {
    if (!is_numeric($vect[$i]))
```

```
      trigger_error("Value at position $i is not a number, using 0 (zero)",
        WARNING);
    $temp[$i] = log($scale) * $vect[$i];
  }
  return $temp;
}

// configura il gestore dell’errore definito dall’utente
$old_error_handler = set_error_handler("myErrorHandler");

// attiva alcuni errori, definendo prima un array misto con elementi non numerici
echo "vector a\n";
$a = array(2,3,"foo",5.5,43.3,21.11);
print_r($a);

// genera il secondo array, generando un avviso
echo "----\nvector b - a warning (b = log(PI) * a)\n";
$b = scale_by_log($a, M_PI);
print_r($b);

// questo è il problema, passiamo una stringa al posto di un array
echo "----\nvector c - an error\n";
$c = scale_by_log("not array",2.3);
var_dump($c);

// errore critico, il log di zero o di un numero negativo non è definito
echo "----\nvector d - fatal error\n";
$d = scale_by_log($a, -2.5);

?>
```

E l’esecuzione di questo script, darà

```
vector a
Array
(
    [0] => 2
    [1] => 3
    [2] => foo
    [3] => 5.5
    [4] => 43.3
    [5] => 21.11
)
----
vector b - a warning (b = log(PI) * a)
<b>WARNING</b> [1024] Value at position 2 is not a number, using 0 (zero)<br>
Array
(
    [0] => 2.2894597716988
    [1] => 3.4341896575482
    [2] => 0
    [3] => 6.2960143721717
    [4] => 49.566804057279
    [5] => 24.165247890281
```

```
)
----
vector c - an error
<b>ERROR</b> [512] Incorrect input vector, array of values expected<br>
NULL
----
vector d - fatal error
<b>FATAL</b> [256] log(x) for x <= 0 is undefined, you used: scale = -2.5<br>
  Fatal error in line 36 of file trigger_error.php, PHP 4.0.2 (Linux)<br>
Aborting...<br>
```

E' importante ricordare che il gestore degli errori standard di PHP viene completamente saltato. La configurazione di error_reporting() non avrà effetto e il vostro gestore di errore sarà richiamato senza considerarla - in ogni caso sarà possibile leggere il valore corrente di error_reporting() ed agire di conseguenza. E' di particolare rilevanza il fatto che questo valore sarà 0 se la riga che causa l'errore viene precedura dall' operatore di controllo errore @.

Notare anche che è vostra responsabilità definire die() se necessario. Se la funzione di gestione dell'errore ritorna, lo script continerà l'esecuzione con la riga seguente a quella che ha causato l'errore.

Vedere anche error_reporting(), restore_error_handler(), trigger_error(), user_error()

trigger_error (PHP 4)

Genera un messaggio a livello utente di errore/avviso/avvertimento message

void **trigger_error** (string error_msg [, int error_type]) \linebreak

Utilizzata per attivare una condizione di errore utente, può essere usata in congiunzione con il gestore di errore interno, o con una funzione definita dall'utente che sia configurata per essere il nuovo gestore di errore con (set_error_handler()). Funziona soltanto con la famiglia di costanti E_USER, e punta alla predefinita `E_USER_NOTICE`.

Questa funzione è utile quando sia necessario generare una particolare risposta ad un'eccezione durante l'esecuzione. Per esempio:

```
if (assert ($divisor == 0))
   trigger_error ("Cannot divide by zero", E_USER_ERROR);
```

Nota: Vedere set_error_handler() per un esempio più esplicativo.

Vedere anche error_reporting(), set_error_handler(), restore_error_handler(), user_error()

user_error (PHP 4 >= 4.0.0)

Genera un messaggio di errore/avviso/avvertimento a livello utente

void **user_error** (string error_msg [, int error_type]) \linebreak

Questo è un alias per la funzione trigger_error().

Vedere anche error_reporting(), set_error_handler(), restore_error_handler() e trigger_error()

XXVIII. FrontBase Functions

Queste funzioni permettono di accedere ai servers del database FrontBase. Affinché queste funzioni siano disponibili è necessario compilare php con il supporto fbsql usando l' opzione `--with-fbsql`.Se si usa questa opzione senza specificare il percorso a fbsql, php cercherà le librerie client di fbsql nella cartella di default specificata nell'istallazione di FrontBase, a seconda del sistema operativo. Se si installa FrontBase in una cartella non standard è necessario specificare sempre il percorso a fbsql: `--with-fbsql=/path/to/fbsql`. In questo modo si forzerà php ad usare le librerie client installate da FrontBase, evitando ogni conflitto.

Maggiori informazioni su FrontBase: http://www.frontbase.com/.

Documentazione su FrontBase : http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation.

Il supporto Frontbase è stato aggiunto dal PHP 4.0.6.

fbsql_affected_rows (PHP 4 >= 4.0.6)

Restituisce il numero di righe (tuple) interessate nella precedente operazione di FrontBase

int **fbsql_affected_rows** ([int link_identifier]) \linebreak

fbsql_affected_rows() restituisce il numero di righe interessate dall'ultima query INSERT, UPDATE or DELETE associata al parametro *`link_identifier`*. Se tale parametro non è stato specificato, sarà usata l'ultima connessione aperta da fbsql_connect().

> **Nota:** Se si stanno usando le transazioni, è necessario chiamare la funzione **fbsql_affected_rows()** dopo una query INSERT, UPDATE, or DELETE, non dopo la chiusura della transazione (commit).

Se l'ultima query è un istruzione DELETE senza clausola WHERE, tutte le righe verranno cancellate dalla tabella e la funzione restituirà il valore 0 (zero).

> **Nota:** Se si utilizza l'istruzione UPDATE, FrontBase non aggiornerà le colonne in cui il valore nuovo è uguale a quello vecchio. Quindi esiste la possibilità che **fbsql_affected_rows()** sia diverso dal numero di righe realmente interessate dalla query.

Se l'ultima query fallisce la funzione restituisce -1.

Vedere anche: fbsql_num_rows().

fbsql_autocommit (PHP 4 >= 4.0.6)

Abilita o disabilita autocommit

bool **fbsql_autocommit** (resource link_identifier [, bool OnOff]) \linebreak

fbsql_autocommit() restituisce lo stato corrente di autocommit. Se è stato specificato il parametro opzionale OnOff, lo stato di autocommit verrà cambiato. Impostando il parametro OnOff su `TRUE` ogni istruzione verrà eseguita automaticamente, in caso di assenza di errori. Impostandolo su `FALSE` l'utente dovrà eseguire la transazione richiamando le funzioni fbsql_commit() o fbsql_rollback().

Vedere anche: fbsql_commit() e fbsql_rollback()

fbsql_change_user (unknown)

Cambia l'identità dell'utente connesso con una connessione attiva

resource **fbsql_change_user** (string user, string password [, string database [, int link_identifier]]) \linebreak

fbsql_change_user() Cambia l'identità dell'utente connesso con una connessione attiva, o con la connessione specificata dal parametro opzionale link_identifier. Se è stato specificato un database, dopo che l'identità dell'utente sarà stata cambiata, questo diventerà il database attivo . Se l'autorizzazione del nuovo utente fallisce, rimarrà attiva l'identità dell'utente corrente.

fbsql_close (PHP 4 >= 4.0.6)

Chiude la connessione a FrontBase

boolean **fbsql_close** ([resource link_identifier]) \linebreak

Restituisce: TRUE in caso di successo, FALSE in caso di fallimento.

fbsql_close() chiude la connessione al sever FrontBase associata ad uno specificato link identifier. Se il *link_identifier* non fosse specificato, verrebbe chiusa l'ultima connessione aperta.

Non è sempre necessario usare **fbsql_close()** nel caso di connessioni non permanenti, esse verranno chiuse automaticamente alla fine dell'esecuzione dello script.

Esempio 1. Uso di fbsql_close()

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);
?>
```

Vedere anche: fbsql_connect() e fbsql_pconnect().

fbsql_commit (PHP 4 >= 4.0.6)

Compie una transazione

bool **fbsql_commit** ([resource link_identifier]) \linebreak

restituisce: TRUE in caso di successo, FALSE in caso di fallimento.

fbsql_commit() esegue la transazione corrente scrivendo tutti gli aggiornamenti pendenti, cancella il disco e sblocca tutte le righe della tabella bloccata dalla transazione. Questo comando è necessario solo nel caso in cui autocommit fosse impostato su false.

Vedere anche: fbsql_autocommit() e fbsql_rollback()

fbsql_connect (PHP 4 >= 4.0.6)

Apre una connessione al Server FrontBase

resource **fbsql_connect** ([string hostname [, string username [, string password]]]) \linebreak

Restituisce un valore di link_identifier positivo in caso di successo, o un messaggio di errore in caso di fallimento.

fbsql_connect() Apre una connessione al Server FrontBase.Se i parametri opzionali non sono specificati verranno usati i valori seguenti come default: *hostname* = 'NULL', *username* = '_SYSTEM' e *password* = empty password.

Se si richiamasse, una seconda volta, la funzione **fbsql_connect()** con gli stessi argomenti, non si creerebbe una nuova connessione,ma verrebbe restituito il valore di link_identifier della connessione già aperta.

La connessione al server si chiuderà alla fine dello script, a meno che non venga chiusa in anticipo richiamando esplicitamente la funzione fbsql_close().

Esempio 1. fbsql_connect()

```
<?php

    $link = fbsql_connect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    print ("Connected successfully");
    fbsql_close ($link);

?>
```

Vedere anche fbsql_pconnect() e fbsql_close().

fbsql_create_blob (PHP 4 >= 4.2.0)

Create a BLOB

string **fbsql_create_blob** (string blob_data [, resource link_identifier]) \linebreak

Returns: A resource handle to the newly created blob.

fbsql_create_blob() creates a blob from blob_data. The returned resource handle can be used with insert and update commands to store the blob in the database.

Esempio 1. fbsql_create_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "blobfile.bin";
    $fp = fopen($filename, "rb");
    $blobdata = fread($fp, filesize($filename));
    fclose($fp);

    $blobHandle = fbsql_create_blob($blobdata, $link);

    $sql = "INSERT INTO BLOB_TABLE (BLOB_COLUMN) VALUES ($blobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_clob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_clob (PHP 4 >= 4.2.0)

Create a CLOB

string **fbsql_create_clob** (string clob_data [, resource link_identifier]) \linebreak

Returns: A resource handle to the newly created CLOB.

fbsql_create_clob() creates a clob from clob_data. The returned resource handle can be used with insert and update commands to store the clob in the database.

Esempio 1. fbsql_create_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $filename = "clob_file.txt";
    $fp = fopen($filename, "rb");
    $clobdata = fread($fp, filesize($filename));
    fclose($fp);

    $clobHandle = fbsql_create_clob($clobdata, $link);

    $sql = "INSERT INTO CLOB_TABLE (CLOB_COLUMN) VALUES ($clobHandle);";
    $rs = fbsql_query($sql, $link);
?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_create_db (PHP 4 >= 4.0.6)

Crea un database

bool **fbsql_create_db** (string database name [, resource link_identifier]) \linebreak

fbsql_create_db() crea un nuovo database FrontBase sul server, identificato dal parametro link_identifier.

Esempio 1. fbsql_create_db()

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    if (fbsql_create_db ("my_db")) {
```

```
        print("Database created successfully\n");
    } else {
        printf("Error creating database: %s\n", fbsql_error ());
    }
?>
```

Vedere anche fbsql_drop_db().

fbsql_data_seek (PHP 4 >= 4.0.6)

Sposta il puntatore del risultato interno

bool **fbsql_data_seek** (resource result_identifier, int row_number) \linebreak

Restituisce: `TRUE` in caso di successo, `FALSE` in caso di fallimento.

fbsql_data_seek() sposta il puntatore interno al risultato FrontBase associato con uno specificato indice in modo che punti ad un numero di riga specificata . La chiamata successiva alla funzionefbsql_fetch_row() restituirà la riga richiesta.

`row_number` comincia a contare da 0.

Esempio 1. fbsql_data_seek()

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");

    fbsql_select_db ("samp_db")
        or die ("Could not select database");

    $query = "SELECT last_name, first_name FROM friends;";
    $result = fbsql_query ($query)
        or die ("Query failed");

    # fetch rows in reverse order

    for ($i = fbsql_num_rows ($result) - 1; $i >=0; $i--) {
        if (!fbsql_data_seek ($result, $i)) {
            printf ("Cannot seek to row %d\n", $i);
            continue;
        }

        if(!($row = fbsql_fetch_object ($result)))
            continue;

        printf("%s %s<BR>\n", $row->last_name, $row->first_name);
    }

    fbsql_free_result ($result);
?>
```

fbsql_database (PHP 4 >= 4.0.6)

Get or set the database name used with a connection

string **fbsql_database** (resource link_identifier [, string database]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

fbsql_database_password (PHP 4 >= 4.0.6)

Imposta o ricerca la password di un database FrontBase.

string **fbsql_database_password** (resource link_identifier [, string database_password]) \linebreak

Restituisce: La password del database identificato dal parametro link_identifier.

fbsql_database_password() imposta e ricerca la password del database corrente. Se il secondo parametro, opzionale (database_password), è stato specificato la funzione imposta, sul server, il valore del parametro come password del database identificato dal parametro link_identifier. Se il link_identifier non è specificato, verrà utilizzata l'ultima connessione aperta. Se nessuna connessione è aperta, la funzione tenterà di aprirne una come se la funzione fbsql_connect() fosse chiamata, e userà quella.

Questa funzione non modifica la password nel database e neppure è in grado ti recuperare la password di un database.

Esempio 1. Esempio di fbsql_create_clob()

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    fbsql_database_password($link, "secret db password");
    fbsql_select_db($database, $link);
?>
```

Vedere anche: fbsql_connect(), fbsql_pconnect() e fbsql_select_db().

fbsql_db_query (PHP 4 >= 4.0.6)

Manda una query FrontBase

resource **fbsql_db_query** (string database, string query [, resource link_identifier]) \linebreak

Restituisce: un indice FrontBase positivo come risultato della query, o `FALSE` in caso di errore.

fbsql_db_query() seleziona un database ed esegue la query su di esso.Se il parametro opzionale link_identifier non è stato specificato, la funzione ne cercherà una già aperta nel FrontBase server, in caso non ne trovasse alcuna aprirà una nuova connessione come se la funzione fbsql_connect() fosse chiamta senza argomenti.

Vedere anche: fbsql_connect().

fbsql_db_status (PHP 4 >= 4.1.0)

Restituisce lo stato di un dato database.

int **fbsql_db_status** (string database_name [, resource link_identifier]) \linebreak

Restituisce: un valore intero con lo stato corrente.

fbsql_db_status() richiede lo stato corrente del database specificato da *`database_name`*. Se il *`link_identifier`* viene omesso verrà usato quello in uso.

Il valore restituito potrà essere uno delle seguenti costanti:

- `FALSE` - L'exec handler del host era invalido. Questo errore si presenta quando la connessione avviene direttamente al database, tramite il link_identifier, usando un numero di porta. FBExec può essere disponibile sul server ma nessuna connessione è stata creata.
- FBSQL_UNKNOWN - Lo stato è sconosciuto.
- FBSQL_STOPPED - Il database non è attivo. Usare fbsql_start_db() per attivare il database.
- FBSQL_STARTING - Il database è in fase di attivazione.
- FBSQL_RUNNING - Il database è attivo e può essere usato per eseguire operazioni SQL.
- FBSQL_STOPPING - Il database é in fase di disattivazione.
- FBSQL_NOEXEC - FBExec non è attivo sul server quindi non è possibile conoscere lo stato del database.

Vedere anche: fbsql_start_db() e fbsql_stop_db().

fbsql_drop_db (PHP 4 >= 4.0.6)

Cancella un database FrontBase

bool **fbsql_drop_db** (string database_name [, resource link_identifier]) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

fbsql_drop_db() procede con l’eliminazione di un intero database dal server associato all’identificatore di link.

fbsql_errno (PHP 4 >= 4.0.6)

Ritorna il valore numerico del messaggio di errore emesso dalla precedente operazione FrontBase.

int **fbsql_errno** ([resource link_identifier]) \linebreak

Restituisce il numero di errore dell’ultima funzione fbsql, `0` (zero) se non ci sono stati errori.

Gli errori vengono restituiti dal database fbsql senza visualizzare i warnings. Usare **fbsql_errno()** per ricevere codice dell’errore. Da notare che questa funzione restituisce solamente il codice di errore proveniente dalla più recente esecuzione di una funzione fbsql (escludendo fbsql_error() e **fbsql_errno()**),così se si vuole usarla, assicurarsi di controllare il valore prima di richiamare un’altra funzione fbsql.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

Vedere anche: fbsql_error() e fbsql_warnings().

fbsql_error (PHP 4 >= 4.0.6)

Ritorna il testo del messaggio di errore emesso dalla precedente operazione FrontBase.

string **fbsql_error** ([resource link_identifier]) \linebreak

Restituisce il testo del messaggio di errore dell’ultima funzione fbsql, o `''` (una stringa vuota) se non ci sono stati errori.

Gli errori vengono restituiti dal database fbsql senza visualizzare i warnings. Usare fbsql_errno() per ricevere codice dell’errore. Da notare che questa funzione restituisce solamente il codice di errore proveniente dalla più recente esecuzione di una funzione fbsql (escludendo **fbsql_error()** e fbsql_errno()),così se si vuole usarla, assicurarsi di controllare il valore prima di richiamare un’altra funzione fbsql.

```
<?php
fbsql_connect("marliesle");
echo fbsql_errno().": ".fbsql_error()."<BR>";
```

```
fbsql_select_db("nonexistentdb");
echo fbsql_errno().": ".fbsql_error()."<BR>";
$conn = fbsql_query("SELECT * FROM nonexistenttable;");
echo fbsql_errno().": ".fbsql_error()."<BR>";
?>
```

Vedere anche: fbsql_errno() e fbsql_warnings().

fbsql_fetch_array (PHP 4 >= 4.0.6)

Restituisce una riga (tupla) di risultato in forma di Array associativo, Array enumerato o entrambi

array **fbsql_fetch_array** (resource result [, int result_type]) \linebreak

Restituisce un array che corrisponde alla riga di risultato, o `FALSE` se non ci sono righe successive.

fbsql_fetch_array() è una versione estesa di fbsql_fetch_row(). In aggiunta all'inserimento dei dati negli elementi dell'array con indice numerico, li inserisce anche in indici associativi, usando il nome dei campi come chiavi.

Se due o più colonne di risultato hanno lo stesso nome di campo , l'ultima colonna sovrascriverà la precedente con lo stesso nome. Per accedere alle altre colonne con lo stesso nome si deve usare l'indice numerico oppure fare un alias della colonna.

```
select t1.f1 as foo t2.f1 as bar from t1, t2
```

Una cosa importante da notare è che **fbsql_fetch_array()** NON è singnificativamente più lenta di fbsql_fetch_row(), mentre fornisce un significativo valore aggiunto.

Il secondo parametro opzionale, *`result_type`* in **fbsql_fetch_array()** è una costante che può assumere i seguenti valori: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

Per ulteriori dettagli vedere anche fbsql_fetch_row() e fbsql_fetch_assoc().

Esempio 1. fbsql_fetch_array()

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select user_id, fullname from table");
while ($row = fbsql_fetch_array ($result)) {
    echo "user_id: ".$row["user_id"]."<br>\n";
    echo "user_id: ".$row[0]."<br>\n";
    echo "fullname: ".$row["fullname"]."<br>\n";
    echo "fullname: ".$row[1]."<br>\n";
}
fbsql_free_result ($result);
```

```
?>
```

fbsql_fetch_assoc (PHP 4 >= 4.0.6)

Restituisce una riga (tupla) di risultato in forma di Array associativo.

array **fbsql_fetch_assoc** (resource result) \linebreak

Restituisce un array associativo corrispondente alla riga di risultato, o FALSE se non ci sono righe successive.

fbsql_fetch_assoc() è equivalente ad una chiamata a fbsql_fetch_array() con FBSQL_ASSOC come parametro opzionale. Restituirà solo un array associativo. fbsql_fetch_array() originariamente lavora in questo modo. Se si vuole un indice numerico come pure quello associativo, usare fbsql_fetch_array().

Se due o più colonne di risultato hanno lo stesso nome di campo , l'ultima colonna sovrascriverà la precedente con lo stesso nome. Per accedere alle altre colonne con lo stesso nome si deve usare fbsql_fetch_array() che ritorna un indice numerico.

Una cosa importante da notare è che **fbsql_fetch_assoc()** NON è singnificativamente più lenta di fbsql_fetch_row(), mentre fornisce un significativo valore aggiunto.

Per maggiori dettagli, vedi anche fbsql_fetch_row() e fbsql_fetch_array().

Esempio 1. fbsql_fetch_assoc()

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database","select * from table");
while ($row = fbsql_fetch_assoc ($result)) {
    echo $row["user_id"];
    echo $row["fullname"];
}
fbsql_free_result ($result);
?>
```

fbsql_fetch_field (PHP 4 >= 4.0.6)

Get column information from a result and return as an object

object **fbsql_fetch_field** (resource result [, int field_offset]) \linebreak

Returns an object containing field information.

fbsql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **fbsql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be `NULL`
- type - the type of the column

Esempio 1. fbsql_fetch_field() example

```
<?php
fbsql_connect ($host, $user, $password)
    or die ("Could not connect");
$result = fbsql_db_query ("database", "select * from table")
    or die ("Query failed");
# get column metadata
$i = 0;
while ($i < fbsql_num_fields ($result)) {
    echo "Information for column $i:<BR>\n";
    $meta = fbsql_fetch_field ($result);
    if (!$meta) {
        echo "No information available<BR>\n";
    }
    echo "<PRE>
max_length:   $meta->max_length
name:         $meta->name
not_null:     $meta->not_null
table:        $meta->table
type:         $meta->type
</PRE>";
    $i++;
}
fbsql_free_result ($result);
?>
```

See also fbsql_field_seek().

fbsql_fetch_lengths (PHP 4 >= 4.0.6)

Get the length of each output in a result

array **fbsql_fetch_lengths** ([resource result]) \linebreak

Returns: An array that corresponds to the lengths of each field in the last row fetched by fbsql_fetch_row(), or FALSE on error.

fbsql_fetch_lengths() stores the lengths of each result column in the last row returned by fbsql_fetch_row(), fbsql_fetch_array() and fbsql_fetch_object() in an array, starting at offset 0.

See also: fbsql_fetch_row().

fbsql_fetch_object (PHP 4 >= 4.0.6)

Fetch a result row as an object

object **fbsql_fetch_object** (resource result [, int result_type]) \linebreak

Returns an object with properties that correspond to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_object() is similar to fbsql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: FBSQL_ASSOC, FBSQL_NUM, and FBSQL_BOTH.

Speed-wise, the function is identical to fbsql_fetch_array(), and almost as quick as fbsql_fetch_row() (the difference is insignificant).

Esempio 1. fbsql_fetch_object() example

```
<?php
fbsql_connect ($host, $user, $password);
$result = fbsql_db_query ("database", "select * from table");
while ($row = fbsql_fetch_object ($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
fbsql_free_result ($result);
?>
```

See also: fbsql_fetch_array() and fbsql_fetch_row().

fbsql_fetch_row (PHP 4 >= 4.0.6)

Get a result row as an enumerated array

array **fbsql_fetch_row** (resource result) \linebreak

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

fbsql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **fbsql_fetch_row()** would return the next row in the result set, or `FALSE` if there are no more rows.

See also: fbsql_fetch_array(), fbsql_fetch_object(), fbsql_data_seek(), fbsql_fetch_lengths(), and fbsql_result().

fbsql_field_flags (PHP 4 >= 4.0.6)

Get the flags associated with the specified field in a result

string **fbsql_field_flags** (resource result, int field_offset) \linebreak

fbsql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using explode().

fbsql_field_len (PHP 4 >= 4.0.6)

Returns the length of the specified field

int **fbsql_field_len** (resource result, int field_offset) \linebreak

fbsql_field_len() returns the length of the specified field.

fbsql_field_name (PHP 4 >= 4.0.6)

Get the name of the specified field in a result

string **fbsql_field_name** (resource result, int field_index) \linebreak

fbsql_field_name() returns the name of the specified field index. *`result`* must be a valid result identifier and *`field_index`* is the numerical offset of the field.

Nota: *`field_index`* starts at 0.

e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Esempio 1. fbsql_field_name() example

```
// The users table consists of three fields:
//   user_id
//   username
```

```
//   password.

$res = fbsql_db_query("users", "select * from users", $link);

echo fbsql_field_name($res, 0) . "\n";
echo fbsql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

fbsql_field_seek (PHP 4 >= 4.0.6)

Set result pointer to a specified field offset

bool **fbsql_field_seek** (resource result, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to fbsql_fetch_field() doesn't include a field offset, the field offset specified in **fbsql_field_seek()** will be returned.

See also: fbsql_fetch_field().

fbsql_field_table (PHP 4 >= 4.0.6)

Get name of the table the specified field is in

string **fbsql_field_table** (resource result, int field_offset) \linebreak

Returns the name of the table that the specified field is in.

fbsql_field_type (PHP 4 >= 4.0.6)

Get the type of the specified field in a result

string **fbsql_field_type** (resource result, int field_offset) \linebreak

fbsql_field_type() is similar to the fbsql_field_name() function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as

detailed in the FrontBase documentation (http://www.frontbase.com/cgi-bin/WebObjects/FrontBase.woa/wa/productsPage?currentPage=Documentation).

Esempio 1. fbsql_field_type() example

```
<?php

fbsql_connect ("localhost", "_SYSTEM", "");
fbsql_select_db ("wisconsin");
$result = fbsql_query ("SELECT * FROM onek;");
$fields = fbsql_num_fields ($result);
$rows   = fbsql_num_rows ($result);
$i = 0;
$table = fbsql_field_table ($result, $i);
echo "Your '".$table."' table has ".$fields." fields and ".$rows." records <BR>";
echo "The table has the following fields <BR>";
while ($i < $fields) {
    $type  = fbsql_field_type  ($result, $i);
    $name  = fbsql_field_name  ($result, $i);
    $len   = fbsql_field_len   ($result, $i);
    $flags = fbsql_field_flags ($result, $i);
    echo $type." ".$name." ".$len." ".$flags."<BR>";
    $i++;
}
fbsql_close();

?>
```

fbsql_free_result (PHP 4 >= 4.0.6)

Free result memory

bool **fbsql_free_result** (int result) \linebreak

fbsql_free_result() will free all memory associated with the result identifier `result`.

fbsql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

fbsql_get_autostart_info (PHP 4 >= 4.1.0)

No description given yet

array **fbsql_get_autostart_info** ([resource link_identifier]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

fbsql_hostname (PHP 4 >= 4.0.6)

Get or set the host name used with a connection

string **fbsql_hostname** (resource link_identifier [, string host_name]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

fbsql_insert_id (PHP 4 >= 4.0.6)

Get the id generated from the previous INSERT operation

int **fbsql_insert_id** ([resource link_identifier]) \linebreak

fbsql_insert_id() returns the ID generated for an column defined as DEFAULT UNIQUE by the previous INSERT query using the given *`link_identifier`*. If *`link_identifier`* isn't specified, the last opened link is assumed.

fbsql_insert_id() returns 0 if the previous query does not generate an DEFAULT UNIQUE value. If you need to save the value for later, be sure to call fbsql_insert_id() immediately after the query that generates the value.

> **Nota:** The value of the FrontBase SQL function `LAST_INSERT_ID()` always contains the most recently generated DEFAULT UNIQUE value, and is not reset between queries.

fbsql_list_dbs (PHP 4 >= 4.0.6)

List databases available on a FrontBase server

resource **fbsql_list_dbs** ([resource link_identifier]) \linebreak

fbsql_list_dbs() will return a result pointer containing the databases available from the current fbsql daemon. Use the fbsql_tablename() function to traverse this result pointer.

Esempio 1. fbsql_list_dbs() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');
$db_list = fbsql_list_dbs($link);

while ($row = fbsql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
```

The above example would produce the following output:

```
database1
database2
database3
...
```

> **Nota:** The above code would just as easily work with fbsql_fetch_row() or other similar functions.

fbsql_list_fields (PHP 4 >= 4.0.6)

List FrontBase result fields

resource **fbsql_list_fields** (string database_name, string table_name [, resource link_identifier]) \linebreak

fbsql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with fbsql_field_flags(), fbsql_field_len(), fbsql_field_name(), and fbsql_field_type().

A result identifier is a positive integer. The function returns -1 if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@fbsql()` then this error string will also be printed out.

Esempio 1. fbsql_list_fields() example

```
$link = fbsql_connect('localhost', 'myname', 'secret');

$fields = fbsql_list_fields("database1", "table1", $link);
$columns = fbsql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo fbsql_field_name($fields, $i) . "\n";;
```

```
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

fbsql_list_tables (PHP 4 >= 4.0.6)

List tables in a FrontBase database

resource **fbsql_list_tables** (string database [, resource link_identifier]) \linebreak

fbsql_list_tables() takes a database name and returns a result pointer much like the fbsql_db_query() function. The fbsql_tablename() function should be used to extract the actual table names from the result pointer.

fbsql_next_result (PHP 4 >= 4.0.6)

Move the internal result pointer to the next result

bool **fbsql_next_result** (int result_id) \linebreak

When sending more than one SQL statement to the server or executing a stored procedure with multiple results will cause the server to return multiple result sets. This function will test for additional results available form the server. If an additional result set exists it will free the existing result set and prepare to fetch the words from the new result set. The function will return `TRUE` if an additional result set was available or `FALSE` otherwise.

Esempio 1. fbsql_next_result() example

```
<?php
    $link = fbsql_connect ("localhost", "_SYSTEM", "secret");
    fbsql_select_db("MyDB", $link);
    $SQL = "Select * from table1; select * from table2;";
    $rs = fbsql_query($SQL, $link);
    do {
        while ($row = fbsql_fetch_row($rs)) {
        }
    } while (fbsql_next_result($rs));
```

```
    fbsql_free_result($rs);
    fbsql_close ($link);
?>
```

fbsql_num_fields (PHP 4 >= 4.0.6)

Get number of fields in result

int **fbsql_num_fields** (resource result) \linebreak

fbsql_num_fields() returns the number of fields in a result set.

See also: fbsql_db_query(), fbsql_query(), fbsql_fetch_field(), and fbsql_num_rows().

fbsql_num_rows (PHP 4 >= 4.0.6)

Get number of rows in result

int **fbsql_num_rows** (resource result) \linebreak

fbsql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows returned from a INSERT, UPDATE or DELETE query, use fbsql_affected_rows().

Esempio 1. fbsql_num_rows() example

```
<?php

$link = fbsql_connect("localhost", "username", "password");
fbsql_select_db("database", $link);

$result = fbsql_query("SELECT * FROM table1;", $link);
$num_rows = fbsql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

See also: fbsql_affected_rows(), fbsql_connect(), fbsql_select_db(), and fbsql_query().

fbsql_password (PHP 4 >= 4.0.6)

Get or set the user password used with a connection

string **fbsql_password** (resource link_identifier [, string password]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

fbsql_pconnect (PHP 4 >= 4.0.6)

Open a persistent connection to a FrontBase Server

resource **fbsql_pconnect** ([string hostname [, string username [, string password]]]) \linebreak

Returns: A positive FrontBase persistent link identifier on success, or `FALSE` on error.

fbsql_pconnect() establishes a connection to a FrontBase server. The following defaults are assumed for missing optional parameters: *`host`* = 'localhost', *`username`* = "_SYSTEM" and *`password`* = empty password.

fbsql_pconnect() acts very much like fbsql_connect() with two major differences.

To set Frontbase server port number, use fbsql_select_db().

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use.

This type of links is therefore called 'persistent'.

fbsql_query (PHP 4 >= 4.0.6)

Send a FrontBase query

resource **fbsql_query** (string query [, resource link_identifier]) \linebreak

fbsql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if fbsql_connect() was called with no arguments, and use it.

> **Nota:** The query string shall always end with a semicolon.

fbsql_query() returns TRUE (non-zero) or FALSE to indicate whether or not the query succeeded. A return value of TRUE means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **fbsql_query()** fails and returns FALSE:

Esempio 1. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT * WHERE 1=1")
    or die ("Invalid query");
?>
```

The following query is semantically invalid if my_col is not a column in the table my_tbl, so **fbsql_query()** fails and returns FALSE:

Esempio 2. fbsql_query() example

```
<?php
$result = fbsql_query ("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

fbsql_query() will also fail and return FALSE if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call fbsql_num_rows() to find out how many rows were returned for a SELECT statement or fbsql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

For SELECT statements, **fbsql_query()** returns a new result identifier that you can pass to fbsql_result(). When you are done with the result set, you can free the resources associated with it by calling fbsql_free_result(). Although, the memory will automatically be freed at the end of the script's execution.

See also: fbsql_affected_rows(), fbsql_db_query(), fbsql_free_result(), fbsql_result(), fbsql_select_db(), and fbsql_connect().

fbsql_read_blob (PHP 4 >= 4.2.0)

Read a BLOB from the database

string **fbsql_read_blob** (string blob_handle [, resource link_identifier]) \linebreak

Returns: A string containing the BLOB specified by blob_handle.

fbsql_read_blob() reads BLOB data from the database. If a select statement contains BLOB and/or BLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_blob()** to get the actual BLOB data from the database.

Esempio 1. fbsql_read_blob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT BLOB_COLUMN FROM BLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain the blob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the BLOB data in the first row
    $blob_data = fbsql_read_blob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), **fbsql_read_blob()**, fbsql_read_clob(), and fbsql_set_lob_mode().

fbsql_read_clob (PHP 4 >= 4.2.0)

Read a CLOB from the database

string **fbsql_read_clob** (string clob_handle [, resource link_identifier]) \linebreak

Returns: A string containing the CLOB specified by clob_handle.

fbsql_read_clob() reads CLOB data from the database. If a select statement contains BLOB and/or CLOB columns FrontBase will return the data directly when data is fetched. This default behavior can be changed with fbsql_set_lob_mode() so the fetch functions will return handles to BLOB and CLOB data. If a handle is fetched a user must call **fbsql_read_clob()** to get the actual CLOB data from the database.

Esempio 1. fbsql_read_clob() example

```
<?php
    $link = fbsql_pconnect ("localhost", "_SYSTEM", "secret")
        or die ("Could not connect");
    $sql = "SELECT CLOB_COLUMN FROM CLOB_TABLE;";
    $rs = fbsql_query($sql, $link);
    $row_data = fbsql_fetch_row($rs);
```

```
    // $row_data[0] will now contain the clob data for teh first row
    fbsql_free_result($rs);

    $rs = fbsql_query($sql, $link);
    fbsql_set_lob_mode($rs, FBSQL_LOB_HANDLE);
    $row_data = fbsql_fetch_row($rs);
    // $row_data[0] will now contain a handle to the CLOB data in the first row
    $clob_data = fbsql_read_clob($row_data[0]);
    fbsql_free_result($rs);

?>
```

See also: fbsql_create_blob(), fbsql_read_blob(), **fbsql_read_clob()**, and fbsql_set_lob_mode().

fbsql_result (PHP 4 >= 4.0.6)

Get result data

mixed **fbsql_result** (resource result, int row [, mixed field]) \linebreak

fbsql_result() returns the contents of one cell from a FrontBase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tabledname.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **fbsql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **fbsql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: fbsql_fetch_row(), fbsql_fetch_array(), and fbsql_fetch_object().

fbsql_rollback (PHP 4 >= 4.0.6)

Rollback a transaction to the database

bool **fbsql_rollback** ([resource link_identifier]) \linebreak

Returns: `TRUE` on success, `FALSE` on failure.

fbsql_rollback() ends the current transaction by rolling back all statements issued since last commit. This command is only needed if autocommit is set to false.

See also: fbsql_autocommit() and fbsql_commit()

fbsql_select_db (PHP 4 >= 4.0.6)

Select a FrontBase database

bool **fbsql_select_db** (string database_name [, resource link_identifier]) \linebreak

Returns: TRUE on success, FALSE on error.

fbsql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if fbsql_connect() was called, and use it.

The client contacts FBExec to obtain the port number to use for the connection to the database. If the database name is a number the system will use that as a port number and it will not ask FBExec for the port number. The FrontBase server can be stared as FRontBase -FBExec=No -port=<port number> <database name>.

Every subsequent call to fbsql_query() will be made on the active database.

if the database is protected with a database password, the user must call fbsql_database_password() before selecting the database.

See also: fbsql_connect(), fbsql_pconnect(), fbsql_database_password() and fbsql_query().

fbsql_set_lob_mode (PHP 4 >= 4.2.0)

Set the LOB retrieve mode for a FrontBase result set

bool **fbsql_set_lob_mode** (resource result, string database_name) \linebreak

Returns: TRUE on success, FALSE on error.

fbsql_set_lob_mode() sets the mode for retrieving LOB data from the database. When BLOB and CLOB data is stored in FrontBase it can be stored direct or indirect. Direct stored LOB data will always be fetched no matter the setting of the lob mode. If the LOB data is less than 512 bytes it will always be stored directly.

- FBSQL_LOB_DIRECT - LOB data is retrieved directly. When data is fetched from the database with fbsql_fetch_row(), and other fetch functions, all CLOB and BLOB columns will be returned as ordinary columns. This is the default value on a new FrontBase result.
- FBSQL_LOB_HANDLE - LOB data is retrieved as handles to the data. When data is fetched from the database with **fbsql_fetch_row ()**, and other fetch functions, LOB data will be returned as a handle to the data if the data is stored indirect or the data if it is stored direct. If a handle is returned it will be a 27 byte string formatted as "@'000000000000000000000000'".

See also: fbsql_create_blob(), fbsql_create_clob(), fbsql_read_blob(), and fbsql_read_clob().

fbsql_set_transaction (PHP 4 >= 4.2.0)

Set the transaction locking and isolation

void **fbsql_set_transaction** (resource link_identifier, int Locking, int Isolation) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

fbsql_start_db (PHP 4 >= 4.0.6)

Start a database on local or remote server

bool **fbsql_start_db** (string database_name [, resource link_identifier]) \linebreak

Returns: TRUE on success, FALSE on failure.

fbsql_start_db()

See also: fbsql_db_status() and fbsql_stop_db().

fbsql_stop_db (PHP 4 >= 4.0.6)

Stop a database on local or remote server

bool **fbsql_stop_db** (string database_name [, resource link_identifier]) \linebreak

Returns: TRUE on success, FALSE on failure.

fbsql_stop_db()

See also: fbsql_db_status() and fbsql_start_db().

fbsql_tablename (PHP 4 >= 4.2.0)

Get table name of field

string **fbsql_tablename** (resource result, int i) \linebreak

fbsql_tablename() takes a result pointer returned by the fbsql_list_tables() function as well as an integer index and returns the name of a table. The fbsql_num_rows() function may be used to determine the number of tables in the result pointer.

Esempio 1. fbsql_tablename() example

```
<?php
fbsql_connect ("localhost", "_SYSTEM", "");
$result = fbsql_list_tables ("wisconsin");
$i = 0;
```

```
while ($i < fbsql_num_rows ($result)) {
    $tb_names[$i] = fbsql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

fbsql_username (PHP 4 >= 4.0.6)

Get or set the host user used with a connection

string **fbsql_username** (resource link_identifier [, string username]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

fbsql_warnings (PHP 4 >= 4.0.6)

Enable or disable FrontBase warnings

bool **fbsql_warnings** ([bool OnOff]) \linebreak

Returns `TRUE` if warnings is turned on otherwise `FALSE`.

fbsql_warnings() enables or disables FrontBase warnings.

XXIX. Funzioni filePro

Queste funzioni permettono l'accesso in sola lettura dei dati del database filePro.

filePro è un marchio registrato della tecnologia fP Technologies, Inc. Per maggiori informazioni su filePro: http://www.fptech.com/.

filepro (PHP 3, PHP 4 >= 4.0.0)

Legge e verifica la mappa del file

bool **filepro** (string directory) \linebreak

Questa funzione legge e verifica la mappa del file, immagazzinando l'indice del campo e le informazioni.

Non viene eseguito nessun locking, si dovrebbe evitare di modificare il database filePro mentre quest'ultimo potrebbe venire aperto in PHP.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali si sta andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

filepro_fieldcount (PHP 3, PHP 4 >= 4.0.0)

Conta quanti campi sono presenti in un database filePro

int **filepro_fieldcount** (void) \linebreak

Restituisce il numero di campi (colonne) di un database filePro aperto.

Vedere anche filepro().

filepro_fieldname (PHP 3, PHP 4 >= 4.0.0)

Restituisce il nome di un campo

string **filepro_fieldname** (int field_number) \linebreak

Restituisce il nome del campo riferito all'indice inserito come parametro *`field_number`*.

filepro_fieldtype (PHP 3, PHP 4 >= 4.0.0)

Restituisce il tipo del campo

string **filepro_fieldtype** (int field_number) \linebreak

Restituisce il tipo del campo riferito all'indice inserito come parametro *`field_number`*.

filepro_fieldwidth (PHP 3, PHP 4 >= 4.0.0)

Restituisce la dimensione di un campo

int **filepro_fieldwidth** (int field_number) \linebreak

Restituisce la lunghezza del campo riferito all'indice inserito come parametro *field_number*.

filepro_retrieve (PHP 3, PHP 4 >= 4.0.0)

Preleva i dati da un database filePro

string **filepro_retrieve** (int row_number, int field_number) \linebreak

Restituisce dei dati da una locazione specificata nel database.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali si sta andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

filepro_rowcount (PHP 3, PHP 4 >= 4.0.0)

Conta quante righe sono presenti in un database filePro

int **filepro_rowcount** (void) \linebreak

Restituisce il numero di righe presenti in un database filePro.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali si sta andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

Vedede anche filepro().

XXX. Funzioni sul filesystem

For related functions, see also the Directory e Program Execution sections.

basename (PHP 3, PHP 4 >= 4.0.0)

Restituisce il nome del file dal percorso indicato

string **basename** (string path [, string suffix]) \linebreak

Data una stringa contenente il percorso di un file, questa funzione restituisce il nome del file. Se il nome del file finisce in *suffix* quest'ultimo verrà tagliato.

Su Windows, sia gli slash (`/`) che i backslash (`\`) vengono utilizzati come carattere si separazione nei percorsi. In altri ambienti, si usa solo lo slash semplice (`/`).

Esempio 1. Esempio di basename()

```
$path = "/home/httpd/html/index.php";
$file = basename ($path);        // la variabile $file contiene "index.php"
$file = basename ($path,".php"); // la variabile $file contiene "index"
```

Nota: Il parametro *suffix* è stato aggiunto in PHP 4.1.0.

Vedere anche: dirname()

chgrp (PHP 3, PHP 4 >= 4.0.0)

Cambia il gruppo del file

int **chgrp** (string filename, mixed group) \linebreak

Tenta di cambiare il gruppo del file *filename* in *group* (specificato per nome o numero). Solo l'amministratore può cambiare arbitrariamente il gruppo di un file; gli altri utenti possono cambiare il gruppo dei file solo fra gruppi di cui sono membri.

Restituisce TRUE se ha successo; FALSE altrimenti.

Vedere anche chown() e chmod().

Nota: Questa funzione non è implementata su piattaforme Windows

chmod (PHP 3, PHP 4 >= 4.0.0)

Cambia le impostazioni del file

int **chmod** (string filename, int mode) \linebreak

Tenta di cambiare le impostazioni del file *`filename`* in quelle date in *`mode`*.

Si osservi che *`mode`* non viene automaticamente assunto come valore ottale, per cui le stringhe (come "g+w") non verranno elaborate correttamente. Per ottenere l'operazione desiderata, è necessario far iniziare *`mode`* con uno zero (0):

```
chmod ("/somedir/somefile", 755);   // decimale; probabilmente errato
chmod ("/somedir/somefile", "u+rwx,go+rx"); // stringa; errato
chmod ("/somedir/somefile", 0755);  // ottale; valore corretto di mode
```

Restituisce `TRUE` se ha successo; `FALSE` altrimenti.

Vedere anche chown() e chgrp().

> **Nota:** Questa funzione non è implementata su piattaforme Windows

chown (PHP 3, PHP 4 >= 4.0.0)

Cambia il proprietario del file

int **chown** (string filename, mixed user) \linebreak

Tenta di cambiare il proprietario di un file *`filename`* in *`user`* (specificato per nome o numero). Solo l'amministratore può cambiare il proprietario di un file.

Restituisce `TRUE` se ha successo; `FALSE` altrimenti.

Vedere anche **chown()** e chmod().

> **Nota:** Questa funzione non è implementata su piattaforme Windows

clearstatcache (PHP 3, PHP 4 >= 4.0.0)

Libera la cache dallo stat

void **clearstatcache** (void) \linebreak

Invocare le funzioni di sistema stat o lstat comporta un buon dispendio di risorse nella maggioranza dei sistemi. Tuttavia, il risultato del'ultima chiamata ad una qualunque funzione di stato (elencate sotto) viene memorizzato per utilizzarlo sulla successiva chiamata analoga allo stesso file. Se intendi forzare una nuova verifica di stato, per esempio perchè il file è stato verificato molte volte e può cambiare o sparire, puoi usare questa funzione per liberare la memoria dai risultati dell'ultima chiamata.

Tale valore viene memorizzato solo per la durata di una singola richiesta.

Le funzioni coinvolte sono stat(), lstat(), file_exists(), is_writable(), is_readable(), is_executable(), is_file(), is_dir(), is_link(), filectime(), fileatime(), filemtime(), fileinode(), filegroup(), fileowner(), filesize(), filetype() e fileperms().

copy (PHP 3, PHP 4 >= 4.0.0)

Copia un file

int **copy** (string source, string dest) \linebreak

Copia un file. Restituisce `TRUE` se la copia riesce, `FALSE` altrimenti.

Esempio 1. copy() example

```
if (!copy($file, $file.'.bak')) {
    print ("failed to copy $file...&lt;br>\n");
}
```

Nota: As of PHP 4.3.0, both *`source`* and *`dest`* may be URLs if the "fopen wrappers" have been enable. See fopen() for more details.

Attenzione

If the destination file already exists, it will be overwritten.

See also move_uploaded_file(), rename(), and the section of the manual about handling file uploads.

delete (unknown)

Una funzione che non c'è

void **delete** (string file) \linebreak

Questa funzione non esiste, la sua presenza nel manuale vuole essere utile a coloro che cercano nel posto sbagliato le funzioni unlink() o unset().

Vedere anche: unlink() per cancellare file, unset() per eliminare variabili.

dirname (PHP 3, PHP 4 >= 4.0.0)

Restituisce il nome della directory dal percorso indicato

string **dirname** (string path) \linebreak

Data una stringa contenente il percorso di un file, questa funzione restituirà il nome della directory.

Su windows sia gli slash (`/`) che i backslash (`\`) vengono utilizzati come caratteri di separazione nei percorsi. In altri ambienti, c'è solo lo slash in avanti (`/`).

Esempio 1. dirname() example

```
$path = "/etc/passwd";
$file = dirname ($path); // $file contiene "/etc"
```

> **Nota:** In PHP 4.0.3, **dirname()** was fixed to be POSIX-compliant. Essentially, this means that if there are no slashes in *`path`* , a dot ('`.`') is returned, indicating the current directory. Otherwise, the returned string is *`path`* with any trailing `/component` removed. Note that this means that you will often get a slash or a dot back from **dirname()** in situations where the older functionality would have given you the empty string.

Vedere anche: basename()

disk_free_space (PHP 4 >= 4.1.0)

Returns available space in directory

float **disk_free_space** (string directory) \linebreak

Given a string containing a directory, this function will return the number of bytes available on the corresponding filesystem or disk partition.

Esempio 1. disk_free_space() example

```
$df = disk_free_space("/"); // $df contains the number of bytes
                            // available on "/"
```

disk_total_space (PHP 4 >= 4.1.0)

Restituisce lo spazio totale di una directory

float **disk_total_space** (string directory) \linebreak

Data una stringa contenente una directory, questa funzione restituirà il numero totale di byte del filesystem o della partizione corrispondente.

Esempio 1. disk_total_space() example

```
$df = disk_total_space("/"); // $df contiene il numero totale di
                             // byte disponibile su "/"
```

diskfreespace (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Alias of disk_free_space()

float **diskfreespace** (string directory) \linebreak

This is a deprecated alias of disk_free_space(). Use that function instead.

fclose (PHP 3, PHP 4 >= 4.0.0)

Chiude un puntatore a file aperto

bool **fclose** (int fp) \linebreak

Chiude il file puntato da *fp*.

Restituisce TRUE in caso di successo e FALSE altrimenti.

Il puntatore al file deve essere valido e deve puntare ad un file aperto correttamente da fopen() o da fsockopen().

feof (PHP 3, PHP 4 >= 4.0.0)

Verifica se è stata raggiunta la fine del file su un puntatore a file

int **feof** (int fp) \linebreak

Restituisce TRUE se il puntatore al file ha raggiunto la fine del file (EOF) o si è verificato un errore; altrimenti restituisce FALSE.

Il puntatore al file deve essere valido e deve puntare ad un file correttamente aperto da fopen(), popen() o fsockopen().

fflush (PHP 4)

Invia l'output in un file

int **fflush** (int fp) \linebreak

Questa funzione forza la scrittura di tutto l'output bufferizzato sulla risorsa puntata da puntatore *fp*. Restituisce TRUE se ha successo, FALSE altrimenti.

Il puntatore al file deve essere valido e deve puntare ad un file correttamente aperto da fopen(), popen() o fsockopen().

fgetc (PHP 3, PHP 4 >= 4.0.0)

Prende un carattere da un puntatore a file

string **fgetc** (int fp) \linebreak

Restituisce una stringa contenente un singolo carattere letto dal file puntato da fp. Restituisce FALSE alla fine del file (EOF).

Il puntatore al file deve essere valido e deve puntare ad un file correttamente aperto da fopen(), popen() o fsockopen().

Vedere anche fread(), fopen(), popen(), fsockopen() e fgets().

fgetcsv (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Prende una riga da un puntatore a file e l'analizza in cerca di campi CSV

array **fgetcsv** (int fp, int lunghezza [, string delimitatore]) \linebreak

Simile a fgets() eccetto per il fatto che **fgetcsv()** analizza le righe lette alla ricerca di campi in in formato CSV e restituisce un vettore contenente i campi letti. Il delimitatore di campo è una virgola, a meno che non venga specificato un altro delimitatore usando il terzo parametro opzionale.

fp deve essere un puntatore valido ad un file correttamente aperto da fopen(), popen() o fsockopen().

lunghezza deve essere maggiore della linea più lunga trovata nel file CSV (compresi i caratteri di fine riga).

fgetcsv() restituisce FALSE in caso d'errore e al raggiungimento della fine del file.

> **Nota:** Una riga vuota in un file CVS verrà riportata come un vettore contenente un solo campo vuoto (NULL) e non verrà trattata come un errore.

Esempio 1. Esempio di fgetcsv() - Legge e scrive l'intero contenuto di un file CSV.

```
$row = 1;
$fp = fopen ("test.csv","r");
while ($data = fgetcsv ($fp, 1000, ",")) {
    $num = count ($data);
    print "<p> $num campi sulla linea $row: <br>";
    $row++;
    for ($c=0; $c < $num; $c++) {
        print $data[$c] . "<br>";
    }
}
fclose ($fp);
```

fgets (PHP 3, PHP 4 >= 4.0.0)

Prende una riga da un puntatore a file

string **fgets** (int fp, int length) \linebreak

Restituisce una stringa di length - 1 byte letti dal file puntato da fp. La lettura termina quando sono stati letti length - 1 byte, in un caratere i newline (che viene incluso nel valore restituito), o alla fine del file (EOF) qualora giunga prima. Se no length is specified, the length defaults to 1k, or 1024 bytes.

Se si verifica un errore, restituisce FALSE.

Errori comuni:

Le persone abituate alla semantica 'C' di fgets notino la differenza nel trattamento dell'EOF.

Il puntatore al file deve essere valido e deve puntare ad un file correttamente aperto da fopen(), popen(), o fsockopen().

Segue un semplice esempio:

Esempio 1. Legge un file riga per riga

```
$fd = fopen ("/tmp/inputfile.txt", "r");
while (!feof ($fd)) {
    $buffer = fgets($fd, 4096);
    echo $buffer;
}
fclose ($fd);
```

Nota: The *length* parameter became optional in PHP 4.2.0

Vedere anche fread(), fopen(), popen(), fgetc(), fsockopen() e socket_set_timeout().

fgetss (PHP 3, PHP 4 >= 4.0.0)

Prende una riga da un puntatore a file ed elimina i tag HTML

string **fgetss** (int fp, int length [, string allowable_tags]) \linebreak

Identica a fgets(), eccetto per il fatto che fgetss tenta di eliminare tutti i tag HTML e PHP dal testo che legge.

Puoi utilizzare il terzo parametro (opzionale) per specificare quali tag non devono essere eliminati.

> **Nota:** Il parametro *allowable_tags* è stato aggiunto in PHP 3.0.13, PHP4B3.

Vedere anche fgets(), fopen(), fsockopen(), popen() e strip_tags().

file (PHP 3, PHP 4 >= 4.0.0)

Legge l'intero file in un vettore

array **file** (string filename [, int use_include_path]) \linebreak

Identica a readfile(), eccetto per il fatto che **file()** restituisce il file in un vettore. Ogni elemento del vettore corrisponde ad una riga del file, con il carattere di newline ancora inserito.

> **Nota:** Each line in the resulting array will include the line ending, so you still need to use trim() if you do not want the line ending present.

Puoi impostare il secondo parametro (opzionale) ad "1", se vuoi cercare il file nel include_path.

```
<?php
// inserisce una pagina web in un array e la stampa
$fcontents = file ('http://www.php.net');
while (list ($line_num, $line) = each ($fcontents)) {
    echo "&lt;b&gt;Line $line_num:&lt;/b&gt; " . htmlspecialchars ($line) . "&lt;br&gt;\n
}

// inserisce una pagina web in una stringa
$fcontents = join ('', file ('http://www.php.net'));
?>
```

Nota: As of PHP 4.3.0 you can use file_get_contents() to return the contents of a file as a string in a binary safe manner.

Attenzione

Questa funzione non è (ancora) binary-safe! (non gestisce correttamente i file binari)

Suggerimento: È possibile utilizzare una URL come un nome di file con questa funzione se è stata abilitata l'opzione "fopen wrappers". Per maggiori informazioni vedere fopen().

Vedere anche readfile(), fopen(), fsockopen() e popen().

file_exists (PHP 3, PHP 4 >= 4.0.0)

Controlla se un file esiste

bool **file_exists** (string filename) \linebreak

Restituisce `TRUE` se il file specificato da *`filename`* esiste; `FALSE` altrimenti.

Questa funzione non lavora con i file remoti; il file da esaminare deve essere accessibile attraverso il filesystem del server.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

file_get_contents (PHP 4 CVS only)

Reads entire file into a string

string **file_get_contents** (string filename [, int use_include_path]) \linebreak

Identical to readfile(), except that **file_get_contents()** returns the file in a string.

Nota: Questa funzione è binary-safe (gestisce correttamente i file binari)

Suggerimento: È possibile utilizzare una URL come un nome di file con questa funzione se è stata abilitata l'opzione "fopen wrappers". Per maggiori informazioni vedere fopen().

file_get_wrapper_data (PHP 4 CVS only)

Retrieves header/meta data from "wrapped" file pointers

mixed **file_get_wrapper_data** (int fp) \linebreak

This function returns header or meta data from files opened with fopen(). This is useful to return the response headers for HTTP connections, or some other statistics for other resources.

The format of the returned data is deliberately undocumented at this time, and depends on which wrapper(s) were used to open the file.

> **Nota:** This function was introduced in PHP 4.3.0.

file_register_wrapper (PHP 4 CVS only)

Register a URL wrapper implemented as a PHP class

boolean **file_register_wrapper** (string protocol, string classname) \linebreak

This function is currently only documented by the example below:

Esempio 1. Implementing a base64 encoding protocol

```
class Base64EncodingStream {
    var $fp = null;

    function stream_open($path, $mode, $options, &$opened_path)
    {
        $this->fp = fopen($path, $mode);
        return is_resource($this->fp);
    }
    function stream_close()
    {
        fclose($this->fp);
    }
    function stream_read($count)
    {
        return false; // We only allow writing
    }
    function stream_write($data)
    {
        return fwrite($this->fp, base64_encode($data));
    }
    function stream_flush()
    {
        fflush($this->fp);
        return true;
    }
    function stream_seek($offset, $whence)
    {
```

```
        return false;
    }
    function stream_gets()
    {
        return false;
    }
    function stream_tell()
    {
        return false;
    }
    function stream_eof()
    {
        return false;
    }
}
file_register_wrapper("base64", "Base64EncodingStream")
    or die("Failed to register protocol");

copy("/tmp/inputfile.txt", "base64:///tmp/outputfile.txt");
readfile("/tmp/outputfile");
```

file_register_wrapper() will return false if the `protocol` already has a handler, or if "fopen wrappers" are disabled.

> **Nota:** This function was introduced in PHP 4.3.0.

fileatime (PHP 3, PHP 4 >= 4.0.0)

Prende l'ora dell'ultimo accesso al file

int **fileatime** (string filename) \linebreak

Restituisce l'ora in cui il file ha ricevuto l'ultimo accesso, o FALSE in caso di errore. L'ora viene restituita come un timestamp Unix.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

N.B.: Si suppone che l'atime di un file cambi ogni volta che i blocchi di dati del file vengono letti. Ciò può risultare costoso per le performance quando una applicazione accede con regolarità ad un numero elevato di file o directory. Alcuni filesystem Unix possono essere montati con l'aggiornamento dell'atime disabilitato per aumentare le performance di tali applicazioni; Gli spool delle news USENET costituiscono un esempo frequente. In tali filesystem queste funzioni sono inutili.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

filectime (PHP 3, PHP 4 >= 4.0.0)

Prende l'ora in cui l'inode del file è stato modificato

int **filectime** (string filename) \linebreak

Restituisce l'ora in cui il file è stato cambiato l'ultima volta o `FALSE` in caso d'errore. L'ora viene restituita come un timestamp di Unix.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Nota: In molti filesystem Unix, si considera un file modificato, quando il suo inode viene cambiato; cioè quando i permessi, il proprietario, il gruppo o altri metadata dell'inode vengono aggiornati. Vedere anche filemtime() (che è ciò che ti serve se vuoi inserire la scritta "Ultima modifica: " nel piede delle tue pagine web) e fileatime().

Sappi anche che in alcuni testi su Unix si fa riferimento al ctime di un file come l'ora di creazione dello stesso. E' sbagliato. Nella maggioranza dei filesystem Unix non esiste un oa di creazione.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

filegroup (PHP 3, PHP 4 >= 4.0.0)

Restituisce il gruppo di un file

int **filegroup** (string filename) \linebreak

Restituisce il l'ID del gruppo cui appartiene il proprietario del file, o `FALSE` in caso d'errore. L'ID del gruppo viene restituito in formato numerico: usa posix_getgrgid() per trasformarlo nel nome del gruppo.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

> **Nota:** Questa funzione non è implementata su piattaforme Windows

> **Nota:** Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

fileinode (PHP 3, PHP 4 >= 4.0.0)

Restituisce il numero di inode del file

int **fileinode** (string filename) \linebreak

Restituisce il numero di inode del file, o `FALSE` in caso d'errore.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

> **Nota:** Questa funzione non è implementata su piattaforme Windows

filemtime (PHP 3, PHP 4 >= 4.0.0)

Restituisce l'ora delle modifiche al file

int **filemtime** (string filename) \linebreak

Restituisce l'ora dell'ultima modifica al file o `FALSE` in caso d'errore. L'ora viene restituita come un timestamp di Unix.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

Nota: Questa funzione restituisce l'ora in cui i blocchi di dati di un file vengono scritti, cioè l'ora in cui il contenuto del file è cambiato. Applica date() al risultato di questa funzione per ottenere una data di modifica da utilizzare nei piedi delle tue pagine.

fileowner (PHP 3, PHP 4 >= 4.0.0)

Restituisce il proprietario del file

int **fileowner** (string filename) \linebreak

Restituisce l'ID dell'utente proprietario del file, o `FALSE` in caso di errore. L'ID dell'utente viene restituito in formato numerico, usa posix_getpwuid() per trasformarlo nel nome dell'utente stesso.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

> **Nota:** Questa funzione non è implementata su piattaforme Windows

fileperms (PHP 3, PHP 4 >= 4.0.0)

Restituisce i permessi del file

int **fileperms** (string filename) \linebreak

Restituisce i permessi sul file, o FALSE in caso d'errore.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

filesize (PHP 3, PHP 4 >= 4.0.0)

Restituisce la dimensione del file

int **filesize** (string filename) \linebreak

Restituisce la dimensione di un file, o FALSE in caso d'errore.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

filetype (PHP 3, PHP 4 >= 4.0.0)

Restituisce il tipo di file

string **filetype** (string filename) \linebreak

Restituisce il tipo del file. Sono valori possibili fifo, char, dir, block, link, file e unknown.

Restituisce FALSE se si verifica un errore.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

Vedere anche: is_dir(), is_file(), is_link(), file_exists() e stat().

flock (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Sistema di bloccaggio file

bool **flock** (int fp, int operation [, int wouldblock]) \linebreak

Il PHP supporta un tecnologia portabile per bloccare file completi in modalità advisory (tutti i programmi che vi accedono, devono usare lo stesso tipo di bloccaggio o non funzionerà).

flock() opera su *`fp`* che deve essere un puntatore ad un file aperto. *`operation`* può assumere uno dei valori seguenti:

- Per acquisire una chiave condivisa (in lettura), imposta *`operation`* a LOCK_SH (usa 1 prima di PHP 4.0.1).
- Per acquisire una chiave esclusiva (in scrittura), imposta *`operation`* a LOCK_EX (usa 2 prima di PHP 4.0.1).
- Per rilasciare una chiave (condivisa o esclusiva), imposta *`operation`* a LOCK_UN (usa 3 prima PHP 4.0.1).
- Se non vuoi che **flock()** blocchi mentre, imposta come LOCK_NB (4 prima di PHP 4.0.1) *`operation`*.

flock() ti permette di utilizzare un semplice modello di lettura/scrittura che in teoria può essere usato su qualsiasi piattaforma (inclusi molti sistemi Unix e anche Windows). Il terzo argomento (opzionale) può essere impostato a `TRUE` se la chiave puo bloccare (EWOULDBLOCK errno condition)

flock() restituisce `TRUE` se ha successo e `FALSE` in caso d'errore (ad esempio quando una chiave (lock) non può venire acquisita).

> **Nota:** Because **flock()** requires a file pointer, you may have to use a special lock file to protect access to a file that you intend to truncate by opening it in write mode (with a "w" or "w+" argument to fopen()).

Attenzione

flock() will not work on NFS and many other networked file systems. Check your operating system documentation for more details.

Su molti sistemi operativi **flock()** è implementato a livello di processo. Usando un server API multithread quale ISAPI non potrai basarti su **flock()** per proteggerre i file da altri script PHP che girino in thread paralleli della stessa istanza del server!

fopen (PHP 3, PHP 4 >= 4.0.0)

Apre un file o un URL

int **fopen** (string filename, string mode [, int use_include_path]) \linebreak

Se *`filename`* inizia con "http://" (indipendentemente dalle maiuscole), viene aperta una connessione HTTP 1.0 al server specificato, la pagina viene richiesta usando il metodo HTTP GET e un puntatore a file viene restituito all'inizio del corpo della risposta. Un header 'Host:' viene inviato con la richiesta in modo di soddisfare i virtual host name-based.

As of PHP 4.3.0 (not yet released), if you have compiled in support for OpenSSL, you may use "https://" to open an HTTP connection over SSL.

Nota che il puntatore al file ti permette di ottenere solo il *body* della risposta; to retrieve the HTTP response header you need to be using PHP 4.0.5 or later; The headers will be stored in the $http_response_header variable. As of PHP 4.3.0 (not yet released), the header information can be retrieved using the **fgetwrapperdata()**.

HTTP connections are read-only; you cannot write data or copy files to an HTTP resource.

Le versioni precedenti al PHP 4.0.5 non elaborano i redirect HTTP. Perciò, le directory devono includere gli slash finali.

Se *filename* inizia con "ftp://" (case insensitive), Viene aperta una connessione ftp al server specificato e viene restituito un puntatore al file richiesto. Se il server non supporta l'FTP passivo, fallirà. Via ftp puoi aprire file sia in lettura che scrittura (ma non contemporaneamente).

If *filename* is one of "php://stdin", "php://stdout", or "php://stderr", the corresponding stdio stream will be opened. (This was introduced in PHP 3.0.13; in earlier versions, a filename such as "/dev/stdin" or "/dev/fd/0" must be used to access the stdio streams.)

If *filename* begins with anything else, the file will be opened from the filesystem, and a file pointer to the file opened is returned.

If the open fails, the function returns `FALSE`.

mode può essere uno dei seguenti:

- 'r' - Apre in sola lettura; posiziona il puntatore all'inizio del file.
- 'r+' - Apre in lettura e scrittura; posiziona il puntatore all'inizio del file.
- 'w' - Apre il file in sola scrittura; posiziona il puntatore all'inizio del file e tronce il file alla lunghezza zero. Se il file non esiste, tenta di crearlo.
- 'w+' - Apre in lettura e scrittura; posiziona il puntatore all'inizio del file e tronce il file alla lunghezza zero. Se il file non esiste, tenta di crearlo.
- 'a' - Apre in sola scrittura; posiziona il puntatore alla fine del file. Se il file non esiste, tenta di crearlo.
- 'a+' - Apre in lettura e scrittura; posiziona il puntatore alla fine del file. Se il file non esiste, tenta di crearlo.

> **Nota:** *mode* può anche essere impostato a 'b'. Ciò è utile solo in sistemi che differenziano fra file binari e di testo (ad esempio Windows. è inutile su Unix). Se non è necessario, tale valore verrà ignorato.

Puoi utilizzare il terzo parametro opzionale ed impostarlo ad "1", se vuoi cercare il file anche nel include_path.

Esempio 1. Esempio di fopen()

```
$fp = fopen ("/home/rasmus/file.txt", "r");
$fp = fopen ("/home/rasmus/file.gif", "wb");
$fp = fopen ("http://www.example.com/", "r");
```

```
$fp = fopen ("ftp://user:password@example.com/", "w");
```

Se incontri problemi nella lettura o scrittura di file e stai usando una versione di PHP come modulo server, ricorda di verificare che i file e le directory che stai usando siano accessibili al processo server.

Sulle piattaforme Windows, fai attenzione ad dichiarare ogni backslash utilizzata nel percorso del file, o utilizza gli slash semplici.

```
$fp = fopen ("c:\\data\\info.txt", "r");
```

Vedere anche fclose(), fsockopen(), socket_set_timeout() e popen().

fpassthru (PHP 3, PHP 4 >= 4.0.0)

Invia tutti i dati rimanenti su un puntartore a file

int **fpassthru** (int fp) \linebreak

Legge fino a EOF sul puntatore al file dato e scrive i risultati sullo standard output.

Se si verifica un errore, **fpassthru()** restituisce `FALSE`.

Il puntatore al file deve essere valido e deve puntare ad un file correttamente aperto da fopen(), popen() o fsockopen(). You may need to call rewind() to reset the file pointer to the beginning of the file if you have already written data to the file. Il file viene chiuso quando **fpassthru()** viene conclusa la sua lettura (lasciando *fp* inutilizzato).

Se desideri semplicemente inviare il contenuto di un file sullo stdout, potresti preferire readfile(), che ti salva la chiamata a fopen().

> **Nota:** When using **fpassthru()** on a binary file on Windows systems, you should make sure to open the file in binary mode by appending a `b` to the mode used in the call to fopen().

Vedere anche readfile(), fopen(), popen() e fsockopen()

fputs (PHP 3, PHP 4 >= 4.0.0)

Scrive su un puntatore a file

int **fputs** (int fp, string str [, int length]) \linebreak

fputs() è un alias di fwrite() del tutto identico. Nota che il parametro *length* è opzionale e se non specificato verrà scritta l'intera stringa.

fread (PHP 3, PHP 4 >= 4.0.0)

Legge un file salvaguardando la corrispondenza binaria

string **fread** (int fp, int length) \linebreak

fread() legge fino a *length* byte dal puntatore al file indicato da *fp*. La lettura finisce quando sono stati letti *length* byte o è stata raggiunta EOF, qualora giunga prima.

```
// copia il contenuto di un file in una stringa
$filename = "/usr/local/something.txt";
$fd = fopen ($filename, "r");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

> **Nota:** Sui sistemi che differenziano fra file di testo e binari (ad esempio Windows) il file deve essere aperto con il parametro mode di fopen() impostato a 'b'.

```
$filename = "c:\\files\\somepic.gif";
$fd = fopen ($filename, "rb");
$contents = fread ($fd, filesize ($filename));
fclose ($fd);
```

Vedere anche fwrite(), fopen(), fsockopen(), popen(), fgets(), fgetss(), fscanf(), file() e fpassthru().

fscanf (PHP 4)

Analizza l'input da un file secondo un determinato formato

mixed **fscanf** (int handle, string format [, string var1]) \linebreak

La funzione **fscanf()** è simile a sscanf(), ma prende il proprio input da un file associato con *handle* e interpreta l'input in accordo con il parametro *format*. Se vengono passati solo due parametri a questa funzione, i valori esaminati verranno restituiti in un vettore. Altrimenti, se vengono passati i

parametri opzionali, la funzione restituirà il numero dei valori assegnati. I parametri opzionali devono essere passati da reference.

Esempio 1. Esempio di fscanf()

```
$fp = fopen ("users.txt","r");
while ($userinfo = fscanf ($fp, "%s\t%s\t%s\n")) {
    list ($name, $profession, $countrycode) = $userinfo;
    //... fai quacosa coi valori ...
}
fclose($fp);
```

Esempio 2. users.txt

```
javier  argonaut        pe
hiroshi sculptor        jp
robert  slacker us
luigi   florist it
```

Vedere anche fread(), fgets(), fgetss(), sscanf(), printf() e sprintf().

fseek (PHP 3, PHP 4 >= 4.0.0)

Sposta un puntatore sul file

int **fseek** (int fp, int offset [, int whence]) \linebreak

Imposta l'indicatore di posizione del file riferito da *fp*. La nuova posizione, misurata in byte dall'inizio del file, viene ottenuta aggiungento *offset* alla posizione specificata da *whence*, i cui valori sono definiti come segue:
SEEK_SET - Imposta la posizione uguale a *offset* byte.
SEEK_CUR - Imposta la posizione alla attuale più *offset*.
SEEK_END - Imposta la posizione alla fine del file più *offset*. (To move to a position before the end-of-file, you nee

Se *whence* non viene specificato, viene assunto come SEEK_SET.

In caso di successo, restituisce 0; altrimenti, restituisce -1. Nota che spostarsi oltre EOF non è considerato un errore.

Non può essere usato su puntatori a file restituiti da fopen() se è in uso il formato "http://" o "ftp://".

> **Nota:** L'argomento *whence* è stato aggiunto dopo PHP 4.0.0.

Vedere anche ftell() e rewind().

fstat (PHP 4 >= 4.0.0)

Restituisce le informazioni riguardanti un file attraverso un puntatore al file aperto

array **fstat** (int fp) \linebreak

Restituisce le statistiche del file aperto dal puntatore fp. Questa funzione è simile a stat() eccetto per il fatto che opera su un puntatore a file aperto invece che su un nome di file.

Restituisce un vettore con le statistiche del file con i seguenti elementi:

1. device
2. inode
3. numero di link
4. user id del proprietario
5. group id del proprietario
6. tipo di device se è presente l'inode device *
7. dimensione in byte
8. ora dell'ultimo accesso
9. ora dell'ultima modifica
10. ora del ultimo cambiamento
11. blocksize per l'I/O del filesystem *
12. numero di blocchi allocati

* - valido solo su sistemi che supportano il tipo st_blksize --negli altri sistemi (per esempio Windows) restituisce -1

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

ftell (PHP 3, PHP 4 >= 4.0.0)

Comunica la posizione di lettura/scrittura del puntatore al file

int **ftell** (int fp) \linebreak

Restituisce la posizione del puntatore al file indicato da fp; cioè, il suo offset all'interno del flusso del file.

Se si verifica un errore, restituisce FALSE.

Il puntatore al file deve essere valido e deve puntare ad un file aperto correttamente da fopen() o da popen().

Vedere anche fopen(), popen(), fseek() e rewind().

ftruncate (PHP 4 >= 4.0.0)

Tronca un file alla lunghezza data

int **ftruncate** (int fp, int size) \linebreak

Prende il puntatore al file *fp* e tronca il file alla lunghezza indicata da *size*. Questa funzione restituisce TRUE in caso di successo e FALSE altrimenti.

fwrite (PHP 3, PHP 4 >= 4.0.0)

Scrive un file salvaguardando la corrispondenza binaria

int **fwrite** (int fp, string string [, int length]) \linebreak

fwrite() scrive il contenuto di *string* nel flusso del file puntato da *fp*. Se l'argomento *length* è specificato la scrittura si arresterà dopo aver scritto *length* byte o alla fine di *string* se si verificasse prima.

fwrite() returns the number of bytes written, or -1 on error.

Nota che se il parametro *length* viene specificato, allora l'opzione di configurazione magic_quotes_runtime verrà ignorata e nessuno slash verrà skippato da *string*.

> **Nota:** Su sistemi che differenzino fra file binari e di testo (come Windows) il file deve essere aperto includendo 'b' nel paramentro mode di fopen().

Vedere anche fread(), fopen(), fsockopen(), popen() e fputs().

glob (PHP 4 CVS only)

Find pathnames matching a pattern

array **glob** (string pattern [, int flags]) \linebreak

The **glob()** function searches for all the pathnames matching *pattern* according to the rules used by the shell. No tilde expansion or parameter substitution is done.

Returns an array containing the matched files/directories or FALSE on error.

> **Nota:** This function is disabled in safe mode and therefore will always return FALSE in safe mode.

Esempio 1. Convenient way how glob() can replace opendir() and friends.

```
foreach (glob("*.txt") as $filename) {
    echo "$filename size " . filesize($filename) . "\n";
}
```

This could result in the following output:

```
funclist.txt size 44686
funcsummary.txt size 267625
quickref.txt size 137820
```

See also opendir(), readdir() and closedir().

is_dir (PHP 3, PHP 4 >= 4.0.0)

Dice se la stringa corrisponde ad una directory

bool **is_dir** (string filename) \linebreak

Restituisce `TRUE` se il nome file esiste ed è una directory. Se *`filename`* è un nome file relativo, verrà controllato relativamente alla directory di lavoro in uso.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

Vedere anche chdir(), is_file() e is_link().

is_executable (PHP 3, PHP 4 >= 4.0.0)

Dice se il file indicato è eseguibile

bool **is_executable** (string filename) \linebreak

Restituisce `TRUE` se il filename esiste ed è un eseguibile.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

Vedere anche is_file() e is_link().

is_file (PHP 3, PHP 4 >= 4.0.0)

Dice se il file è un file regolare

bool **is_file** (string filename) \linebreak

Restituisce `TRUE` se il filename esiste ed è un file regolare.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Vedere anche is_dir() e is_link().

is_link (PHP 3, PHP 4 >= 4.0.0)

Dice se il file è un link simbolico

bool **is_link** (string filename) \linebreak

Restituisce TRUE se il filename esiste ed è un link simbolico.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Vedere anche is_dir() e is_file().

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

> **Nota:** Questa funzione non è implementata su piattaforme Windows

is_readable (PHP 3, PHP 4 >= 4.0.0)

Dice se un file è leggibile

bool **is_readable** (string filename) \linebreak

Restituisce TRUE se filename esiste ed è leggibile.

Ricorda che PHP può accedere al file come l'utente che è rappresentato dal server (spesso 'nobody'). Non sono prese in conto limitazione di sicurezza.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

Vedere anche is_writable().

is_uploaded_file (PHP 3>= 3.0.17, PHP 4 >= 4.0.3)

Dice se un file fù caricato via HTTP POST.

bool **is_uploaded_file** (string filename) \linebreak

Returns `TRUE` if the file named by `filename` was uploaded via HTTP POST. This is useful to help ensure that a malicious user hasn't tried to trick the script into working on files upon which it should not be working--for instance, `/etc/passwd`.

This sort of check is especially important if there is any chance that anything done with uploaded files could reveal their contents to the user, or even to other users on the same system.

is_uploaded_file() is available only in versions of PHP 3 after PHP 3.0.16, and in versions of PHP 4 after 4.0.2. If you are stuck using an earlier version, you can use the following function to help protect yourself:

> **Nota:** The following example will *not* work in versions of PHP 4 after 4.0.2. It depends on internal functionality of PHP which changed after that version.

```
<?php
/* Userland test for uploaded file. */
function is_uploaded_file($filename) {
    if (!$tmp_file = get_cfg_var('upload_tmp_dir')) {
        $tmp_file = dirname(tempnam(", "));
    }
    $tmp_file .= '/' . basename($filename);
    /* User might have trailing slash in php.ini... */
    return (ereg_replace('/+', '/', $tmp_file) == $filename);
}

/* This is how to use it, since you also don't have
 * move_uploaded_file() in these older versions: */
if (is_uploaded_file($HTTP_POST_FILES['userfile'])) {
    copy($HTTP_POST_FILES['userfile'], "/place/to/put/uploaded/file");
} else {
    echo "Possible file upload attack: filename '$HTTP_POST_FILES[userfile]'.";
}
?>
```

See also move_uploaded_file(), and the section Handling file uploads for a simple usage example.

is_writable (PHP 4 >= 4.0.0)

Dice se un file è scrivibile

bool **is_writable** (string filename) \linebreak

Restituisce `TRUE` se filename esiste ed è scrivibile. L'argomento filename può essere un nome di directory che ti permetta di verificare se una directory è scrivibile.

Tieni presente che PHP può accedere al file con lo user id del web server (spesso 'nobody'). Non sono prese in considerazioni limitazioni di sicurezza.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

Questa funzione non lavora con i file remoti; i file da esaminare devono essere accessibili attraverso il filesystem del server.

Vedere anche is_readable().

is_writeable (PHP 3, PHP 4 >= 4.0.0)

Dice se un file è scrivibile

bool **is_writeable** (string filename) \linebreak

Questo è un alias per is_writable()

link (PHP 3, PHP 4 >= 4.0.0)

Crea un hard link

int **link** (string target, string link) \linebreak

link() crea un hard link.

Vedere anche symlink() per creare soft link, e readlink() assieme a linkinfo().

> **Nota:** Questa funzione non è implementata su piattaforme Windows

linkinfo (PHP 3, PHP 4 >= 4.0.0)

Restituisce informazioni su un collegamento

int **linkinfo** (string path) \linebreak

linkinfo() restituisce il campo st_dev della struttura stat dello UNIX C restituita dalla chiamata di sistema lstat. Questa funzione è usata per verificare se un link (puntato da *path*) esiste davvero (usando lo stesso metodo della macro S_ISLNK definita in stat.h). Restituisce 0 o FALSE in caso di errore.

Vedere anche symlink(), link() e readlink().

> **Nota:** Questa funzione non è implementata su piattaforme Windows

lstat (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Da informazioni su un file o un link simbolico

array **lstat** (string filename) \linebreak

Raccoglie le statistiche del file o del link simbolico chiamato filename. Questa funzione è identica alla funzione stat() eccetto che se il paramametro `filename` è un link simbolico valido, viene restituito lo stato del link simbolico, non lo stato del file puntato dal link simbolico.

Restituisce un vettore con le statistiche del file con i seguenti elementi:

1. device
2. inode
3. modalità di protezione dell'inode
4. numero di link
5. ID utente del proprietario
6. ID del gruppo del proprietario
7. tipo di device con l'inode device *
8. dimensione in byte
9. ora dell'ultimo aggesso
10. ora del'ultima modifica
11. ora dell'ultimo cambiamento
12. dimensione del blocco per l'I/O di filesystem *
13. numero dei blocchi allocati

* - valida solo su sistemi che supportano il tipo st_blksize--altri sistemi (ad esempio Windows) restituiscono -1.

lstat() cannot be used on remote files.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

mkdir (PHP 3, PHP 4 >= 4.0.0)

Crea una directory

int **mkdir** (string pathname, int mode) \linebreak

Tenta di creare la directory specificata da pathname.

Nota che probabilmente vuoi specificare mode come un ottale, per cui deve iniziare con uno zero. La modalità è anche modificata dall'umask corrente, che puoi cambiare con umask().

```
mkdir ("/path/to/my/dir", 0700);
```

Restituisce `TRUE` se ha successo e `FALSE` se fallisce.

Vedere anche rmdir().

move_uploaded_file (PHP 4 >= 4.0.3)

Muove un file caricato in una nuova posizione

bool **move_uploaded_file** (string filename, string destination) \linebreak

Questa funzione verifica che il file indicato da *filename* è un file validamente caricato (nel senso che è stato caricato attraverso il meccanismo di caricamento HTTP POST di PHP). Se il file è valido, verrà spostato nel file dato da *destination*.

Se *filename* non è un file validamente caricato, allora non verrà compiuta alcuna azione e **move_uploaded_file()** restituirà FALSE.

Se *filename* è un file validamente caricato, ma non può essere mossi per qualche ragione, non verrà compiuto alcunchè e **move_uploaded_file()** restituirà FALSE. In più verrà visualizzato un avviso di pericolo.

Questo tipo di verifica è particolarmente importante se sussiste la possibilità che qualcosa fatto con i file caricati possa rivelare il loro contenuto agli utenti o ad altri utenti dello stesso sistema.

> **Nota:** Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali si sta andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

> **Nota: move_uploaded_file()** is not affected by the normal safe-mode UID-restrictions. This is not unsafe because **move_uploaded_file()** only operates on files uploaded via PHP.

> **Attenzione**
>
> If the destination file already exists, it will be overwritten.

Vedere anche is_uploaded_file() e la sessione Handling file uploads per un semplice esempio.

parse_ini_file (PHP 4 >= 4.0.0)

Parse a configuration file

array **parse_ini_file** (string filename [, bool process_sections]) \linebreak

parse_ini_file() loads in the ini file specified in *filename*, and returns the settings in it in an associative array. By setting the last *process_sections* parameter to TRUE, you get a multidimensional array, with the section names and settings included. The default for *process_sections* is FALSE

> **Nota:** This function has nothing to do with the `php.ini` file. It is already processed, the time you run your script. This function can be used to read in your own application's configuration files.

The structure of the ini file is similar to that of the `php.ini`'s.

Esempio 1. Contents of sample.ini

```
; This is a sample configuration file
; Comments start with ';', as in php.ini

[first_section]
one = 1
five = 5

[second_section]
path = /usr/local/bin
```

Esempio 2. parse_ini_file() example

```
<?php

// Parse without sections
$ini_array = parse_ini_file("sample.ini");
print_r($ini_array);

// Parse with sections
$ini_array = parse_ini_file("sample.ini", TRUE);
print_r($ini_array);

?>
```

Would produce:

```
Array
(
    [one] => 1
    [five] => 5
    [path] => /usr/local/bin
)
Array
(
    [first_section] => Array
        (
            [one] => 1
            [five] => 5
        )

    [second_section] => Array
        (
            [path] => /usr/local/bin
        )
```

)

pathinfo (PHP 4 >= 4.0.3)

Restituisce informazioni su un percorso di file

array **pathinfo** (string path) \linebreak

pathinfo() restituisce un vettore associativo contenente informazioni riguardo *path*. Nel vettore vegono riportati i seguenti elementi: *dirname*, *basename* e *extension*.

Esempio 1. pathinfo() Example

```
<?php

$path_parts = pathinfo("/www/htdocs/index.html");

echo $path_parts["dirname"] . "\n";
echo $path_parts["basename"] . "\n";
echo $path_parts["extension"] . "\n";

?>
```

produrra:

```
/www/htdocs
index.html
html
```

Vedere anche dirname(), basename(), parse_url() e realpath().

pclose (PHP 3, PHP 4 >= 4.0.0)

Chiude un puntatore ad un file di processo

int **pclose** (int fp) \linebreak

Chiude un puntatore ad una pipe aperto da popen().

Il puntatore deve essere valido e deve essere stato restituito da un chiamata a popen() terminata con successo.

Restituisce lo stato di terminazione del processo che stava girando.

Vedere anche popen().

popen (PHP 3, PHP 4 >= 4.0.0)

Apre un puntatore ad un file di processo

int **popen** (string command, string mode) \linebreak

Apre una pipe ad un processo eseguito forzando il comando dato da command.

Restituisce un puntatore a file identico a quello restituito da fopen(), eccetto che per il fatto che è unidirezionale (può solo essere usato per la lettura o la scrittura) e deve essere chiudo con pclose(). Questo puntatore può essere usato con fgets(), fgetss() e fputs().

Se si verifica un errore, restituisce FALSE.

```
$fp = popen ("/bin/ls", "r");
```

Nota: If the command to be executed could not be found, a valid resource is returned. This may seem odd, but makes sense; it allows you to access any error message returned by the shell:

```
<?php
error_reporting(E_ALL);

/* Add redirection so we can get stderr. */
$fp = popen('/path/to/spooge 2>&1', 'r');
echo "'$fp'; " . gettype($fp) . "\n";
$read = fread($fp, 2096);
echo $read;
pclose($fp);
?>
```

Vedere anche pclose().

readfile (PHP 3, PHP 4 >= 4.0.0)

Invia un file

int **readfile** (string filename [, int use_include_path]) \linebreak

Legge un file e lo scrive nello standard output.

Restituisce il numero di byte letti dal file. Se si verifica un errore viene restituito `FALSE` e se la funzione non è stata chiamata come @readfile, un messaggio d'errore verra stampato.

Se *`filename`* inzia con "http://" (case insensitive), viene aperta una connessione HTTP 1.0 al server specificato e il testo della risposta viene scritto nello stadard output.

Versioni precedenti al PHP 4.0.5 non considera le redirezioni HTTP. Perciò, le directory devono includere gli slash finali.

Se *`filename`* inizia con "ftp://" (case insensitive), viene aperta una connessione ftp al server specificato e il file richiesto viene stampato sullo standard output. Se il server non supporta l'ftp passivo, questa funzione fallirà.

Se *`filename`* non inizia con tali stringhe, il file verrà apperto dal filesystem e il suo contenuto verrà scritto nello standard output.

Puoi settare il secondo parametro opzionale ad "1", se vuoi anche cercare il file nella include_path.

Vedere anche fpassthru(), file(), fopen(), include(), require() e virtual().

readlink (PHP 3, PHP 4 >= 4.0.0)

Restituisce il target di un link simbolico

string **readlink** (string path) \linebreak

readlink() si comporta analogamente alla funzione readlink di C e restituisce i conenuti del link simbolico path o 0 in caso d'errore.

Vedere anche symlink(), **readlink()** e linkinfo().

> **Nota:** Questa funzione non è implementata su piattaforme Windows

realpath (PHP 4 >= 4.0.0)

Restituisce un percorso assoluto regolare

string **realpath** (string path) \linebreak

realpath() espande tutti i link simbolici e risolve i riferimenti a '/./', '/../' ed altri caratteri '/' nell'input *`path`* e restituisce il percorso assoluto canonizzato. Il percorso risultante non avrà link simbolici, '/./' o '/../'.

realpath() restituisce `FALSE` in caso di fallimento, per esempio se il file non esiste.

Esempio 1. Esempio di realpath()

```
$real_path = realpath ("../../index.php");
```

Vedere anche: basename(), dirname() e **path_info()**.

rename (PHP 3, PHP 4 >= 4.0.0)

Rinomina un file

int **rename** (string oldname, string newname) \linebreak

Tenta di rinominare *oldname* in *newname*.

Restituisce TRUE se ha successo e FALSE se fallisce.

rewind (PHP 3, PHP 4 >= 4.0.0)

Riavvolge la posizione di un puntatore a file

int **rewind** (int fp) \linebreak

Pone l'indicatore alla posizione del file per fp all'inizio del flusso del file.

Se si verifica un errore, returns 0.

Il puntatore al file deve essere valido e deve puntare ad un file aperto con successo da fopen().

> **Nota:** If you have opened the file in append ("a") mode, any data you write to the file will always be appended, regardless of the file position.

Vedere anche fseek() e ftell().

rmdir (PHP 3, PHP 4 >= 4.0.0)

Rimuove una directory

int **rmdir** (string dirname) \linebreak

Tenta di cancellare la directory indiata da pathname. La directory deve essere vuota e i permessi concessi devono permetterlo. Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

Vedere anche mkdir() e unlink().

set_file_buffer (PHP 3>= 3.0.8, PHP 4)

Imposta il buffering del file sul puntatore al file dato

int **set_file_buffer** (int fp, int buffer) \linebreak

L'output di fwrite() viene normalmente bufferizzato a 8K. Ciò significa che se ci sono due processi che vogliono scrivere nello stesso flusso di output (un file), ciascuno viene interrotto dopo 8K di dati per permettere all'altro di scrivere. **set_file_buffer()** imposta il buffering per le operazioni di scrittura sul dato puntatore a file `fp` a `buffer` byte. Se `buffer` è 0 allora le operazioni di scrittura non vengono bufferizzate. Questo assicura che ogni scrittura di fwrite() venga completata prima che sia permesso ad altri processi di scrivere in quel flusso di output.

La funzione restituisce 0 quando ha successo, o EOF se la richiesta non può essere soddisfatta.

L'esempio seguente mostra come usare **set_file_buffer()** per creare un flusso non bufferizzato.

Esempio 1. set_file_buffer() example

```
$fp=fopen($file, "w");
if($fp){
  set_file_buffer($fp, 0);
  fputs($fp, $output);
  fclose($fp);
}
```

Vedere anche fopen(), fwrite().

stat (PHP 3, PHP 4 >= 4.0.0)

Da informazioni su un file

array **stat** (string filename) \linebreak

Restituisce una serie di informazioni a riguardo del file filename.

Restituisce un vettore con le statistiche del file con i seguenti elementi:

1. device
2. inode
3. modalità di protezione dell'inode
4. numero di link
5. ID utente del proprietario
6. ID del gruppo del proprietario
7. tipo di device con l'inode device *
8. dimensione in byte

9. ora dell'ultimo aggesso
10. ora del'ultima modifica
11. ora dell'ultimo cambiamento
12. dimensione del blocco per l'I/O di filesystem *
13. numero dei blocchi allocati

* - valida solo su sistemi che supportano il tipo st_blksize--altri sistemi (ad esempio Windows) restituiscono -1.

Restituisce `FALSE` in caso d'errore.

stat() cannot be used on remote files.

I risultati di questa funzione vengono memorizzati nella cache. Vedi clearstatcache() per maggiori informazioni.

symlink (PHP 3, PHP 4 >= 4.0.0)

Crea un link simbolico

int **symlink** (string target, string link) \linebreak

symlink() crea un link simbolico dal *`target`* esistente con il nome specificato da *`link`*.

Vedere anche link() per creare hard link, e readlink() con linkinfo().

> **Nota:** Questa funzione non è implementata su piattaforme Windows

tempnam (PHP 3, PHP 4 >= 4.0.0)

Crea un nome di file unico

string **tempnam** (string dir, string prefix) \linebreak

Crea un nome di file temporaneo univoco nella directory specificata. Se la directory non esiste, **tempnam()** può generare il nome file nella directory temporanea del sistema.

Prima di PHP 4.0.6, il comportamento della funzione **tempnam()** dipendeva dal sistema. Su Windows la variabile d'ambiente TMP scavalcherà il parametro *`dir`*, su Linux la variabile d'ambiente TMPDIR ha la precedenza, mentre SVR4 userà sempre il tuo parametro *`dir`* se la directory cui punta esiste. Consulta la documentazione del tuo sistema sulla funzione tempnam(3) se non sei sicuro.

Restituisce il nuovo filename temporaneo, o la stringa `NULL` se fallisce.

Esempio 1. tempnam() example

```
$tmpfname = tempnam ("/tmp", "FOO");
```

```
$fp = fopen($tmpfname, "w");
fwrite($fp, "writing to tempfile");
fclose($fp);

// do here something

unlink($tmpfname);
```

> **Nota:** Il comportamento di questa funzione è cambiata in 4.0.3. Il file temporaneo è anche creato per evitare una condizione di rincorsa dove il file può apparire nel filesystem fra il momento che la stringa viene generato e prima che lo script si occupi di crearlo. Note, that you need to remove the file in case you need it no more, it is not done automatically.

Vedere anche tmpfile() e unlink().

tmpfile (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Crea un file temporaneo

int **tmpfile** (void) \linebreak

Crea un file temporaneo con un nome univoco il modalità di scrittura, restituendo un riferimento al file simile a quello tornato da fopen(). Il file viene automaticamente cancellato una volta chiuso (usando fclose()), o quando lo script termina.

Per dettagli, consulta la documentazione del tuo sistema sulla funzione `tmpfile(3)`, così come il file haeader `stdio.h`.

Esempio 1. tmpfile() example

```
$temp = tmpfile();
fwrite($temp, "writing to tempfile");
fclose($temp); // this removes the file
```

Vedere anche tempnam().

touch (PHP 3, PHP 4 >= 4.0.0)

Imposta l'ora di modifica di un file

int **touch** (string filename [, int time]) \linebreak

Tenta di impostare l'ora di modifica del file indicato da filename al valore dato da time. Se non viene passata l'opzione *time* usa l'ora attuale. This is equivalent to what utime (sometimes referred to as utimes) does.

Se il file non esiste, viene creato.

Restituisce TRUE se ha successo; FALSE altrimenti.

Esempio 1. touch() example

```
if (touch ($FileName)) {
    print "$FileName modification time has been
           changed to todays date and time";
} else {
    print "Sorry Could Not change modification time of $FileName";
}
```

umask (PHP 3, PHP 4 >= 4.0.0)

Cambia l'umask corrente

int **umask** (int mask) \linebreak

umask() imposta l'umask di PHP a & 0777 e restituisce il vecchio umask. Quando PHP è utilizzato come modulo server, l'umask viene ripristinato quando ciascuna richiesta è finita.

umask() senza argomenti restituisce semplicemente l'umask corrente.

> **Nota:** Questa funzione non è implementata su piattaforme Windows

unlink (PHP 3, PHP 4 >= 4.0.0)

Cancella un file

int **unlink** (string filename) \linebreak

Cancella *filename*. Simile alla funzione C di Unix unlink().

Restituisce 0 o FALSE in caso d'errore.

Vedere anche rmdir() per eliminare directory.

XXXI. Forms Data Format functions

Introduzione

Forms Data Format (FDF) is a format for handling forms within PDF documents. You should read the documentation at http://partners.adobe.com/asn/developer/acrosdk/forms.html for more information on what FDF is and how it is used in general.

The general idea of FDF is similar to HTML forms. The difference is basically the format how data is transmitted to the server when the submit button is pressed (this is actually the Form Data Format) and the format of the form itself (which is the Portable Document Format, PDF). Processing the FDF data is one of the features provided by the fdf functions. But there is more. One may as well take an existing PDF form and populated the input fields with data without modifying the form itself. In such a case one would create a FDF document (fdf_create()) set the values of each input field (fdf_set_value()) and associate it with a PDF form (fdf_set_file()). Finally it has to be sent to the browser with MimeType `application/vnd.fdf`. The Acrobat reader plugin of your browser recognizes the MimeType, reads the associated PDF form and fills in the data from the FDF document.

If you look at an FDF-document with a text editor you will find a catalogue object with the name `FDF`. Such an object may contain a number of entries like `Fields`, `F`, `Status` etc.. The most commonly used entries are `Fields` which points to a list of input fields, and `F` which contains the filename of the PDF-document this data belongs to. Those entries are referred to in the FDF documentation as /F-Key or /Status-Key. Modifying this entries is done by functions like fdf_set_file() and fdf_set_status(). Fields are modified with fdf_set_value(), fdf_set_opt() etc..

Requisiti

You must download the FDF toolkit from http://partners.adobe.com/asn/developer/acrosdk/forms.html.

Istallazione

You must compile PHP with `--with-fdftk[=DIR]`.

Nota: If you run into problems configuring PHP with fdftk support, check whether the header file FdfTk.h and the library libFdfTk.so are at the right place. They should be in fdftk-dir/include and

fdftk-dir/lib. This will not be the case if you just unpack the FdfTk distribution.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

Costanti Predefinite

Queste costanti sono definite da questa estensione e sono disponibili solo se l'estensione è stata

compilata nel PHP o se è stata caricata dinamicamente a runtime.

`FDFValue` (integer)

`FDFStatus` (integer)

`FDFFile` (integer)

`FDFID` (integer)

`FDFFf` (integer)

`FDFSetFf` (integer)

`FDFClearFf` (integer)

`FDFFlags` (integer)

`FDFSetF` (integer)

`FDFClrF` (integer)

`FDFAP` (integer)

`FDFAS` (integer)

`FDFAction` (integer)

`FDFAA` (integer)

`FDFAPRef` (integer)

`FDFIF` (integer)

`FDFEnter` (integer)

`FDFExit` (integer)

`FDFDown` (integer)

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
// The pdf form contained several input text fields with the names
// volume, date, comment, publisher, preparer, and two checkboxes
// show_publisher and show_preparer.
$fdf = fdf_open("test.fdf");
$volume = fdf_get_value($fdf, "volume");
echo "The volume field has the value '<B>$volume</B>'<BR>";

$date = fdf_get_value($fdf, "date");
echo "The date field has the value '<B>$date</B>'<BR>";

$comment = fdf_get_value($fdf, "comment");
echo "The comment field has the value '<B>$comment</B>'<BR>";

if(fdf_get_value($fdf, "show_publisher") == "On") {
  $publisher = fdf_get_value($fdf, "publisher");
  echo "The publisher field has the value '<B>$publisher</B>'<BR>";
} else
  echo "Publisher shall not be shown.<BR>";

if(fdf_get_value($fdf, "show_preparer") == "On") {
  $preparer = fdf_get_value($fdf, "preparer");
  echo "The preparer field has the value '<B>$preparer</B>'<BR>";
} else
  echo "Preparer shall not be shown.<BR>";
fdf_close($fdf);
?>
```

fdf_add_template (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Adds a template into the FDF document

bool **fdf_add_template** (int fdfdoc, int newpage, string filename, string template, int rename) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

fdf_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Close an FDF document

bool **fdf_close** (int fdf_document) \linebreak

The **fdf_close()** function closes the FDF document.

See also fdf_open().

fdf_create (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Create a new FDF document

int **fdf_create** (void) \linebreak

The **fdf_create()** creates a new FDF document. This function is needed if one would like to populate input fields in a PDF document with data.

Esempio 1. Populating a PDF document

```
<?php
$outfdf = fdf_create();
fdf_set_value($outfdf, "volume", $volume, 0);

fdf_set_file($outfdf, "http:/testfdf/resultlabel.pdf");
fdf_save($outfdf, "outtest.fdf");
fdf_close($outfdf);
Header("Content-type: application/vnd.fdf");
$fp = fopen("outtest.fdf", "r");
fpassthru($fp);
unlink("outtest.fdf");
?>
```

See also fdf_close(), fdf_save(), fdf_open().

fdf_get_file (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get the value of the /F key

string **fdf_get_file** (int fdf_document) \linebreak

The fdf_set_file() returns the value of the /F key.

See also fdf_set_file().

fdf_get_status (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get the value of the /STATUS key

string **fdf_get_status** (int fdf_document) \linebreak

The **fdf_get_status()** returns the value of the /STATUS key.

See also fdf_set_status().

fdf_get_value (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get the value of a field

string **fdf_get_value** (int fdf_document, string fieldname) \linebreak

The **fdf_get_value()** function returns the value of a field.

See also fdf_set_value().

fdf_next_field_name (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get the next field name

string **fdf_next_field_name** (int fdf_document [, string fieldname]) \linebreak

The **fdf_next_field_name()** function returns the name of the field after the field in `fieldname` or the field name of the first field if the second parameter is `NULL`.

See also fdf_set_value(), fdf_get_value().

fdf_open (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Open a FDF document

int **fdf_open** (string filename) \linebreak

The **fdf_open()** function opens a file with form data. This file must contain the data as returned from a PDF form. Currently, the file has to be created 'manually' by using fopen() and writing the content of HTTP_FDF_DATA with fwrite() into it. A mechanism like for HTML form data where for each input field a variable is created does not exist.

Esempio 1. Accessing the form data

```
<?php
// Save the FDF data into a temp file
$fdffp = fopen("test.fdf", "w");
fwrite($fdffp, $HTTP_FDF_DATA, strlen($HTTP_FDF_DATA));
fclose($fdffp);

// Open temp file and evaluate data
$fdf = fdf_open("test.fdf");
...
fdf_close($fdf);
?>
```

See also fdf_close().

fdf_save (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Save a FDF document

int **fdf_save** (string filename) \linebreak

The **fdf_save()** function saves a FDF document. The FDF Toolkit provides a way to output the document to stdout if the parameter *filename* is '.'. This does not work if PHP is used as an apache module. In such a case one will have to write to a file and use e.g. fpassthru() to output it.

See also fdf_close() and example for fdf_create().

fdf_set_ap (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set the appearance of a field

bool **fdf_set_ap** (int fdf_document, string field_name, int face, string filename, int page_number) \linebreak

The **fdf_set_ap()** function sets the appearance of a field (i.e. the value of the /AP key). The possible values of *face* are 1=FDFNormalAP, 2=FDFRolloverAP, 3=FDFDownAP.

fdf_set_encoding (PHP 4 >= 4.1.0)

Sets FDF character encoding

bool **fdf_set_encoding** (int fdf_document, string encoding) \linebreak

fdf_set_encoding() sets the character encoding in FDF document *fdf_document*. *encoding* should be the valid encoding name. The valid encoding name in Acrobat 5.0 are "`Shift-JIS`", "`UHC`", "`GBK`","`BigFive`".

The **fdf_set_encoding()** is available in PHP 4.1.0 or later.

fdf_set_file (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set the value of the /F key

bool **fdf_set_file** (int fdf_document, string filename) \linebreak

The **fdf_set_file()** sets the value of the /F key. The /F key is just a reference to a PDF form which is to be populated with data. In a web environment it is a URL (e.g. http:/testfdf/resultlabel.pdf).

See also fdf_get_file() and example for fdf_create().

fdf_set_flags (PHP 4 >= 4.0.2)

Sets a flag of a field

bool **fdf_set_flags** (int fdf_document, string fieldname, int whichFlags, int newFlags) \linebreak

The **fdf_set_flags()** sets certain flags of the given field *fieldname*.

See also fdf_set_opt().

fdf_set_javascript_action (PHP 4 >= 4.0.2)

Sets an javascript action of a field

bool **fdf_set_javascript_action** (int fdf_document, string fieldname, int trigger, string script) \linebreak

fdf_set_javascript_action() sets a javascript action for the given field *fieldname*.

See also fdf_set_submit_form_action().

fdf_set_opt (PHP 4 >= 4.0.2)

Sets an option of a field

bool **fdf_set_opt** (int fdf_document, string fieldname, int element, string str1, string str2) \linebreak

The **fdf_set_opt()** sets options of the given field *fieldname*.

See also fdf_set_flags().

fdf_set_status (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set the value of the /STATUS key

bool **fdf_set_status** (int fdf_document, string status) \linebreak

The **fdf_set_status()** sets the value of the /STATUS key.

See also fdf_get_status().

fdf_set_submit_form_action (PHP 4 >= 4.0.2)

Sets a submit form action of a field

bool **fdf_set_submit_form_action** (int fdf_document, string fieldname, int trigger, string script, int flags) \linebreak

The **fdf_set_submit_form_action()** sets a submit form action for the given field *fieldname*.

See also fdf_set_javascript_action().

fdf_set_value (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Set the value of a field

bool **fdf_set_value** (int fdf_document, string fieldname, string value, int isName) \linebreak

The **fdf_set_value()** function sets the value of a field. The last parameter determines if the field value is to be converted to a PDF Name (*isName* = 1) or set to a PDF String (*isName* = 0).

See also fdf_get_value().

XXXII. Funzioni FriBiDi

fribidi_log2vis (PHP 4 >= 4.0.4)

Converte una stringa logica in una visuale

string **fribidi_log2vis** (string str, string direction, int charset) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

XXXIII. Funzioni FTP

Le funzioni in questa estensione implementano l'accesso client ad un file server utilizzando il File Transfer Protocol (FTP) come definito in http://www.faqs.org/rfcs/rfc959.html.

Usando il modulo FTP vengono definite le seguenti costanti: `FTP_ASCII` e `FTP_BINARY`.

Per l'utilizzo delle funzioni FTP con la vostra configurazione PHP, dovrete aggiungere l'opzione `--enable-ftp` durante l'installazione PHP 4, e `--with-ftp` nell'installazione di PHP 3.

Esempio 1. FTP

```
<?php
// stabilire una connessione
$conn_id = ftp_connect($ftp_server);

// login con user name e password
$login_result = ftp_login($conn_id, $ftp_user_name, $ftp_user_pass);

// controllo della connessione
if ((!$conn_id) || (!$login_result)) {
        echo "La connessione FTP è fallita!";
        echo "Tentativo di connessione a $ftp_server per l'utente $ftp_user_name";
        die;
    } else {
        echo "Connesso a $ftp_server, utente $ftp_user_name";
    }

// upload del file
$upload = ftp_put($conn_id, $destination_file, $source_file, FTP_BINARY);

// controllo dello stato di upload
if (!$upload) {
        echo "Il caricamento FTP non è andato a buon fine!";
    } else {
        echo "Caricato il file $source_file su $ftp_server come $destination_file";
    }

// chiudere il flusso FTP
ftp_quit($conn_id);
?>
```

ftp_cdup (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Passa alla directory superiore

int **ftp_cdup** (int ftp_stream) \linebreak

Passa alla directory superiore.

Restituisce TRUE se a buon fine, FALSE in caso di errore.

ftp_chdir (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Cambia le directory su un server FTP

int **ftp_chdir** (int ftp_stream, string directory) \linebreak

Passa alla *directory* specificata.

Restituisce TRUE se a buon fine, FALSE in caso di errore.

ftp_close (PHP 4 >= 4.2.0)

Chiude una connessione FTP

void **ftp_close** (resource ftp_stream) \linebreak

> **Nota:** Questa funzione è disponibile solo nella versione CVS.

ftp_close() chiude *ftp_stream* e rilascia la risorsa. Non si può riutilizzare questa risorsa, ma è necessario crearne una nuova usando ftp_connect().

ftp_connect (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Apre una connessione FTP

resource **ftp_connect** (string host [, int port [, int timeout]]) \linebreak

Restituisce un flusso FTP stream in caso di successo, FALSE in caso di errore.

ftp_connect() Apre una connessione FTP all' *host* specificato. Il parametro *port* specifica un porta alternativa alla quale connettersi. Se omessa oppure impostata a zero, verra' usata la porta 21, default dell'FTP.

Il parametro *timeout* specifica il timeout per tutte le successive operazioni di rete. Se omesso, verrà utilizzato il valore predefinito di 90 secondi. Il timeout può essere cambiato o interrogato in ogni momento usando ftp_set_option() e ftp_get_option().

> **Nota:** Questo parametro è disponibile solo nella versione CVS.

ftp_delete (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Cancella un file sul server FTP

int **ftp_delete** (int ftp_stream, string percorso) \linebreak

ftp_delete() cancella il file specificato da *`percorso`* dal server FTP.

Restituisce `TRUE` se a buon fine, `FALSE` in caso di errore.

ftp_exec (PHP 4 >= 4.0.3)

Richiede l'esecuzione di un programma sul server FTP

bool **ftp_exec** (resource stream, string comando) \linebreak

Invia una richiesta SITE EXEC *`comando`* al server ftp. Restituisce `FALSE` se la richiesta fallisce, altrimenti restituisce l'output del comando.

ftp_fget (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Scarica un file dal server FTP e lo salva in un file aperto

int **ftp_fget** (int ftp_stream, int fp, string remote_file, int mode) \linebreak

ftp_fget() recupera *`remote_file`* dal server FTP, e lo scrive da un dato puntatore del file, *`fp`*. Il parametro *`mode`* che specifica il tipo di trasferimento deve essere `FTP_ASCII` oppure `FTP_BINARY`.

Restituisce `TRUE` se a buon fine, `FALSE` in caso di errore.

ftp_fput (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Carica da un file aperto al server FTP

int **ftp_fput** (int ftp_stream, string remote_file, int fp, int mode) \linebreak

ftp_fput() carica i dati dal puntatore del file *`fp`* fino alla fine del file stesso. I risultati vengono salvati in *`remote_file`* sul server FTP. Il parametro *`mode`* che spedifica il tipo di trasferimento deve essere `FTP_ASCII` oppure `FTP_BINARY`.

Restituisce `TRUE` se a buon fine, `FALSE` in caso di errore.

ftp_get (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Scarica un file dal server FTP

int **ftp_get** (int ftp_stream, string file_in_locale, string file_remoto, int mode) \linebreak

ftp_get() recupera *file_remoto* dal server FTP, e lo salva come *file_in_locale* localmente. La modalità di trasferimento indicata dal parametro *mode* deve essere specificata come `FTP_ASCII` oppure `FTP_BINARY`.

Restituisce `TRUE` se a buon fine, `FALSE` in caso di errore.

Vedere anche ftp_fget().

ftp_get_option (PHP 4 >= 4.2.0)

Richiama vari comportamenti runtime dello stream FTP attualmente in uso

mixed **ftp_get_option** (resource stream, int opzione) \linebreak

Nota: Questa funzione è disponibile solo nella versione CVS.

Restituisce il valore in caso di successo o `FALSE` se la data *opzione* non è supportata. Nel secondo caso viene inoltre generato un messaggio di avviso.

Questa funzione restituisce il valore corrispondente all' *opzione* richiesta, prendendolo dallo *stream* specificato. Al momento sono supportate le seguenti opzioni:

Tabella 1. Opzioni runtime FTP supportate

FTP_TIMEOUT_SEC	Restituisce l'attuale timeout usato per le operazioni relative alla rete.

Esempio 1. Esempio di ftp_get_option()

```
// Ottiene il timeout del dato stream FTP
$timeout = ftp_get_option($conn_id, FTP_TIMEOUT_SEC);
```

ftp_login (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Loggarsi in una connessione FTP

int **ftp_login** (int ftp_stream, string nome_utente, string password) \linebreak

Esegue il login in un dato stream FTP.

Restituisce TRUE se a buon fine, FALSE in caso di errore.

ftp_mdtm (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Restituisce la data di ultima modifica di un dato file

int **ftp_mdtm** (int ftp_stream, string remote_file) \linebreak

ftp_mdtm() Controlla la data di ultima modifica di un file, e la restituisce in formato UNIX timestamp. Se si verifica un errore, oppure il file non esiste, viene restituito -1. Da notare che non tutti i server supportano questa caratteristica.

Restituisce uno UNIX timestamp se a buon fine, oppure -1 in caso di errore.

Nota: ftp_mdtm() non funziona con le cartelle.

ftp_mkdir (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Crea una directory

string **ftp_mkdir** (int ftp_stream, string directory) \linebreak

Crea la *directory* specificata.

Restituisce il nome della directory appena creata se a buon fine, FALSE in caso di errore.

ftp_nlist (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Restituisce l'elenco dei file in una data directory

array **ftp_nlist** (int ftp_stream, string directory) \linebreak

Restituisce una array di nomi di file se a buon fine, FALSE in caso di errore.

ftp_pasv (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Abilita o disabilita la modalità passiva

int **ftp_pasv** (int ftp_stream, int pasv) \linebreak

ftp_pasv() Abilita la modalità passiva se il parametro *pasv* risulta `TRUE` (disabilita la modalità passiva se *pasv* risulta `FALSE`.) In modalità passiva la connessione viene iniziata dal client, piuttosto che dal server.

Restituisce `TRUE` se a buon fine, `FALSE` in caso di errore.

ftp_put (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Carica un file sul server FTP

int **ftp_put** (int ftp_stream, string remote_file, string local_file, int mode) \linebreak

ftp_put() salva *local_file* sul server FTP, come *remote_file*. Il parametro *mode* che spedifica il tipo di trasferimento deve essere `FTP_ASCII` oppure `FTP_BINARY`.

Restituisce `TRUE` se a buon fine, `FALSE` in caso di errore.

Esempio 1. Esempio di ftp_put()

```
$upload = ftp_put ($conn_id, $destination_file, $source_file, FTP_ASCII);
```

ftp_pwd (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Restituisce il nome della directory corrente

string **ftp_pwd** (int ftp_stream) \linebreak

Restituisce il nome della directory corrente, oppure `FALSE` in caso di errore.

ftp_quit (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Chiude una connessione FTP

int **ftp_quit** (int ftp_stream) \linebreak

ftp_quit() chiude *ftp_stream*.

ftp_rawlist (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Restituisce l'elenco dettagliato dei file in una data directory

array **ftp_rawlist** (int ftp_stream, string directory) \linebreak

ftp_rawlist() esegue il comando FTP LIST e restituisce il risultato come un array. Ogni elemento dell'array corrisponde ad una linea di testo. L'output non viene in alcun modo interpretato. L'identificatore del tipo di sistema restituito da ftp_systype() può essere usato per determinare l'interpretazione del risultato.

ftp_rename (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Rinomina un file sul server FTP

boolean **ftp_rename** (resource ftp_stream, string da, string a) \linebreak

ftp_rename() rinomina il file o la cartella che ha come nome corrente *`da`* con il nome *`a`*, sul flusso FTP *`ftp_stream`*.

Restituisce `TRUE` se a buon fine, `FALSE` in caso di errore.

ftp_rmdir (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Elimina una directory

int **ftp_rmdir** (int ftp_stream, string directory) \linebreak

Elimina la *`directory`* specificata.

Restituisce `TRUE` in caso di successo, `FALSE` in caso di errore.

ftp_set_option (PHP 4 >= 4.2.0)

Imposta alcune opzioni runtime generiche dell'FTP

bool **ftp_set_option** (resource stream, int opzione, mixed valore) \linebreak

> **Nota:** Questa funzione è disponibile solo nella versione CVS.

Restituisce `TRUE` se l'opzione può essere impostata; `FALSE` altrimenti. Un messaggio di avviso verrà generato se *`opzione`* non è supportata o se *`valore`* non corrisponde al valore atteso per quella data *`opzione`*.

Questa funzione controlla diverse opzioni runtime per lo stream FTP specificato. Il parametro *`valore`* dipende da quale parametro *`opzione`* si sceglie di alterare. Attualmente, sono supportate le seguenti opzioni:

Tabella 1. Opzioni runtime FTP supportate

FTP_TIMEOUT_SEC	Cambia il timeout in secondi usato per tutte le funzioni orientate alla rete. Il parametro *valore* deve essere di tipo int e deve essere maggiore di 0. Il valore predefinito di timeout è 90 secondi.

Esempio 1. Esempio di ftp_set_option()

```
// Imposta il timeout di rete a 10 secondi
ftp_set_option($conn_id, FTP_TIMEOUT_SEC, 10);
```

ftp_site (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Invia un comando SITE al server

int **ftp_site** (int ftp_stream, string cmd) \linebreak

ftp_site() Invia il comando specificato da *cmd* al server FTP. I comandi SITE non sono standardizzati e cambiano da server a server. Sono utili per gestire i permessi di accesso e di appartenenza ai gruppi dei file.

Restituisce TRUE se a buon fine, FALSE in caso di errore.

ftp_size (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Restituisce la dimensione di un dato file

int **ftp_size** (int ftp_stream, string remote_file) \linebreak

ftp_size() restituisce la dimensione di un file. Se si verifica un errore, oppure se il file non esiste, viene restituito -1. Non tutti i server supportano questa caratteristica.

Restituisce la dimensione del file se a buon fine, oppure -1 in caso di errore.

ftp_systype (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Restituisce l'identificatore del tipo di sistema del server FTP remoto

string **ftp_systype** (int ftp_stream) \linebreak

Restituisce il tipo di sistema remoto, oppure FALSE in caso di errore.

XXXIV. Function Handling functions

Introduzione

These functions all handle various operations involved in working with functions.

Requisiti

Queste funzioni sono disponibili nei moduli standard, che sono sempre disponibili.

Istallazione

Non ‘ necessaria nessuna installazione per usare queste funzioni, esse fanno parte del core di PHP.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Costanti Predefinite

Questa estensione non definisce alcuna costante.

call_user_func (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Call a user function given by the first parameter

mixed **call_user_func** (string function_name [, mixed parameter [, mixed ...]]) \linebreak

Call a user defined function given by the *function_name* parameter. Take the following:

```
function barber ($type) {
    print "You wanted a $type haircut, no problem";
}
call_user_func ('barber', "mushroom");
call_user_func ('barber', "shave");
```

See also: call_user_func_array(), call_user_method(), call_user_method_array().

call_user_func_array (PHP 4 >= 4.0.4)

Call a user function given with an array of parameters

mixed **call_user_func_array** (string function_name [, array paramarr]) \linebreak

Call a user defined function given by *function_name*, with the parameters in *paramarr*. For example:

```
function debug($var, $val)
    echo "***DEBUGGING\nVARIABLE: $var\nVALUE:";
    if (is_array($val) || is_object($val) || is_resource($val))
        print_r($val);
    else
        echo "\n$val\n";
    echo "***\n";
}

$c = mysql_connect();
$host = $HTTP_SERVER_VARS["SERVER_NAME"];

call_user_func_array ('debug', array("host", $host));
call_user_func_array ('debug', array("c", $c));
call_user_func_array ('debug', array("HTTP_POST_VARS", $HTTP_POST_VARS));
```

See also: call_user_func(), call_user_method(), call_user_method_array().

Nota: This function was added to the CVS code after release of PHP 4.0.4pl1

create_function (PHP 4)

Create an anonymous (lambda-style) function

string **create_function** (string args, string code) \linebreak

Creates an anonymous function from the parameters passed, and returns a unique name for it. Usually the *args* will be passed as a single quote delimited string, and this is also recommended for the *code*. The reason for using single quoted strings, is to protect the variable names from parsing, otherwise, if you use double quotes there will be a need to escape the variable names, e.g. `\$avar`.

You can use this function, to (for example) create a function from information gathered at run time:

Esempio 1. Creating an anonymous function with create_function()

```
$newfunc = create_function('$a,$b','return "ln($a) + ln($b) = ".log($a * $b);');
echo "New anonymous function: $newfunc\n";
echo $newfunc(2,M_E)."\n";
// outputs
// New anonymous function: lambda_1
// ln(2) + ln(2.718281828459) = 1.6931471805599
```

Or, perhaps to have general handler function that can apply a set of operations to a list of parameters:

Esempio 2. Making a general processing function with create_function()

```
function process($var1, $var2, $farr) {
    for ($f=0; $f < count($farr); $f++)
    echo $farr[$f]($var1,$var2)."\n";
}

// create a bunch of math functions
$f1 = 'if ($a >=0) {return "b*a^2 = ".$b*sqrt($a);} else {return false;}';
$f2 = "return \"min(b^2+a, a^2,b) = \".min(\$a*\$a+\$b,\$b*\$b+\$a);";
$f3 = 'if ($a > 0 && $b != 0) {return "ln(a)/b = ".log($a)/$b; } else { re-
turn false; }';
$farr = array(
    create_function('$x,$y', 'return "some trig: ".(sin($x) + $x*cos($y));'),
    create_function('$x,$y', 'return "a hypotenuse: ".sqrt($x*$x + $y*$y);'),
    create_function('$a,$b', $f1),
    create_function('$a,$b', $f2),
    create_function('$a,$b', $f3)
    );

echo "\nUsing the first array of anonymous functions\n";
echo "parameters: 2.3445, M_PI\n";
process(2.3445, M_PI, $farr);

// now make a bunch of string processing functions
$garr = array(
    create_function('$b,$a','if (strncmp($a,$b,3) == 0) return "** \"$a\" '.
```

```
    ’and \"$b\"\n** Look the same to me! (looking at the first 3 chars)";’),
    create_function(’$a,$b’,’; return "CRCs: ".crc32($a)." , ".crc32(b);’),
    create_function(’$a,$b’,’; return "similar(a,b) = ".similar_text($a,$b,&$p)."($p%)";’
    );
echo "\nUsing the second array of anonymous functions\n";
process("Twas brilling and the slithy toves", "Twas the night", $garr);
```

and when you run the code above, the output will be:

```
Using the first array of anonymous functions
parameters: 2.3445, M_PI
some trig: -1.6291725057799
a hypotenuse: 3.9199852871011
b*a^2 = 4.8103313314525
min(b^2+a, a^2,b) = 8.6382729035898
ln(a/b) = 0.27122299212594

Using the second array of anonymous functions
** "Twas the night" and "Twas brilling and the slithy toves"
** Look the same to me! (looking at the first 3 chars)
CRCs: -725381282 , 1908338681
similar(a,b) = 11(45.833333333333%)
```

But perhaps the most common use for of lambda-style (anonymous) functions is to create callback functions, for example when using array_walk() or usort()

Esempio 3. Using anonymous functions as callback functions

```
$av = array("the ","a ","that ","this ");
array_walk($av, create_function(’&$v,$k’,’$v = $v."mango";’));
print_r($av);  // for PHP 3 use var_dump()
// outputs:
// Array
// (
//   [0] => the mango
//   [1] => a mango
//   [2] => that mango
//   [3] => this mango
// )

// an array of strings ordered from shorter to longer
$sv = array("small","larger","a big string","it is a string thing");
print_r($sv);
// outputs:
// Array
// (
//   [0] => small
//   [1] => larger
//   [2] => a big string
//   [3] => it is a string thing
```

```
// )

// sort it from longer to shorter
usort($sv, create_function('$a,$b','return strlen($b) - strlen($a);'));
print_r($sv);
// outputs:
// Array
// (
//   [0] => it is a string thing
//   [1] => a big string
//   [2] => larger
//   [3] => small
// )
```

func_get_arg (PHP 4 >= 4.0.0)

Return an item from the argument list

mixed **func_get_arg** (int arg_num) \linebreak

Returns the argument which is at the *arg_num*'th offset into a user-defined function's argument list. Function arguments are counted starting from zero. **func_get_arg()** will generate a warning if called from outside of a function definition.

If *arg_num* is greater than the number of arguments actually passed, a warning will be generated and **func_get_arg()** will return FALSE.

```
<?php
function foo() {
     $numargs = func_num_args();
     echo "Number of arguments: $numargs<br>\n";
     if ($numargs >= 2) {
     echo "Second argument is: " . func_get_arg (1) . "<br>\n";
     }
}

foo (1, 2, 3);
?>
```

func_get_arg() may be used in conjunction with func_num_args() and func_get_args() to allow user-defined functions to accept variable-length argument lists.

> **Nota:** This function was added in PHP 4.

func_get_args (PHP 4 >= 4.0.0)

Returns an array comprising a function's argument list

array **func_get_args** (void) \linebreak

Returns an array in which each element is the corresponding member of the current user-defined function's argument list. **func_get_args()** will generate a warning if called from outside of a function definition.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs<br>\n";
    if ($numargs >= 2) {
    echo "Second argument is: " . func_get_arg (1) . "<br>\n";
    }
    $arg_list = func_get_args();
    for ($i = 0; $i < $numargs; $i++) {
    echo "Argument $i is: " . $arg_list[$i] . "<br>\n";
    }
}

foo (1, 2, 3);
?>
```

func_get_args() may be used in conjunction with func_num_args() and func_get_arg() to allow user-defined functions to accept variable-length argument lists.

> **Nota:** This function was added in PHP 4.

func_num_args (PHP 4 >= 4.0.0)

Returns the number of arguments passed to the function

int **func_num_args** (void) \linebreak

Returns the number of arguments passed into the current user-defined function. **func_num_args()** will generate a warning if called from outside of a user-defined function.

```
<?php
function foo() {
    $numargs = func_num_args();
    echo "Number of arguments: $numargs\n";
}
```

```
foo (1, 2, 3);    // Prints 'Number of arguments: 3'
?>
```

func_num_args() may be used in conjunction with func_get_arg() and func_get_args() to allow user-defined functions to accept variable-length argument lists.

function_exists (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Return `TRUE` if the given function has been defined

bool **function_exists** (string function_name) \linebreak

Checks the list of defined functions, both built-in (internal) and user-defined, for *function_name*. Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

```
if (function_exists('imap_open')) {
    echo "IMAP functions are available.<br>\n";
} else {
    echo "IMAP functions are not available.<br>\n";
}
```

Note that a function name may exist even if the function itself is unusable due to configuration or compiling options (with the image functions being an example). Also note that **function_exists()** will return `FALSE` for constructs, such as include_once() and echo().

See also method_exists() and get_defined_functions().

get_defined_functions (PHP 4 >= 4.0.4)

Returns an array of all defined functions

array **get_defined_functions** (void) \linebreak

This function returns an multidimensional array containing a list of all defined functions, both built-in (internal) and user-defined. The internal functions will be accessible via `$arr["internal"]`, and the user defined ones using `$arr["user"]` (see example below).

```
function myrow($id, $data) {
    return "<tr><th>$id</th><td>$data</td></tr>\n";
}

$arr = get_defined_functions();

print_r($arr);
```

Will output something along the lines of:

```
Array
(
    [internal] => Array
        (
            [0] => zend_version
            [1] => func_num_args
            [2] => func_get_arg
            [3] => func_get_args
            [4] => strlen
            [5] => strcmp
            [6] => strncmp
            ...
            [750] => bcscale
            [751] => bccomp
        )

    [user] => Array
        (
            [0] => myrow
        )

)
```

See also get_defined_vars() and get_defined_constants().

register_shutdown_function (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Register a function for execution on shutdown

int **register_shutdown_function** (string func) \linebreak

Registers the function named by `func` to be executed when script processing is complete.

Multiple calls to **register_shutdown_function()** can be made, and each will be called in the same order as they were registered. If you call exit() within one registered shutdown function, processing will stop completely and no other registered shutdown functions will be called.

The registered shutdown functions are called after the request has been completed (including sending any output buffers), so it is not possible to send output to the browser using echo() or print(), or retrieve the contents of any output buffers using ob_get_contents().

> **Nota:** Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

register_tick_function (PHP 4 >= 4.0.3)

Register a function for execution on each tick

void **register_tick_function** (string func [, mixed arg]) \linebreak

Registers the function named by `func` to be executed when a tick is called.

unregister_tick_function (PHP 4 >= 4.0.3)

De-register a function for execution on each tick

void **unregister_tick_function** (string func [, mixed arg]) \linebreak

De-registers the function named by `func` so it is no longer executed when a tick is called.

XXXV. Gettext

Introduzione

The gettext functions implement a NLS (Native Language Support) API which can be used to internationalize your PHP applications. Please see the gettext documentation from your system for a thorough explanation of these functions or view the docs at http://www.gnu.org/manual/gettext/index.html.

Requisiti

To use this functions you must download and install the GNU gettext package from http://www.gnu.org/software/gettext/gettext.html

Istallazione

To include GNU gettext support in your PHP build you must add the option `--with-gettext[=DIR]` where DIR is the gettext install directory, defaults to /usr/local.

Configurazione Runtime

Questa estensione non definisce alcuna direttiva di configurazione

Resource Type

Questa estensione non definisce alcun tipo di risorsa.

Costanti Predefinite

Questa estensione non definisce alcuna costante.

bind_textdomain_codeset (PHP 4 >= 4.2.0)

Specify the character encoding in which the messages from the DOMAIN message catalog will be returned

string **bind_textdomain_codeset** (string domain, string codeset) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

bindtextdomain (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Sets the path for a domain

string **bindtextdomain** (string domain, string directory) \linebreak

The **bindtextdomain()** function sets the path for a domain.

dcgettext (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Overrides the domain for a single lookup

string **dcgettext** (string domain, string message, int category) \linebreak

This function allows you to override the current domain for a single message lookup. It also allows you to specify a `category`.

dcngettext (PHP 4 >= 4.2.0)

Plural version of dcgettext

string **dcngettext** (string domain, string msgid1, string msgid2, int n, int category) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

dgettext (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Override the current domain

string **dgettext** (string domain, string message) \linebreak

The **dgettext()** function allows you to override the current domain for a single message lookup.

dngettext (PHP 4 >= 4.2.0)

Plural version of dgettext

string **dngettext** (string domain, string msgid1, string msgid2, int n) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

gettext (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Lookup a message in the current domain

string **gettext** (string message) \linebreak

This function returns a translated string if one is found in the translation table, or the submitted message if not found. You may use the underscore character '_' as an alias to this function.

Esempio 1. gettext()-check

```
<?php
// Set language to German
putenv("LANG=de");

// Specify location of translation tables
bindtextdomain("myPHPApp", "./locale");

// Choose domain
textdomain("myPHPApp");

// Print a test message
print gettext("Welcome to My PHP Application");

// Or use the alias _() for gettext()
print _("Have a nice day");
?>
```

ngettext (PHP 4 >= 4.2.0)

Plural version of gettext

string **ngettext** (string msgid1, string msgid2, int n) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

textdomain (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Sets the default domain

string **textdomain** (string text_domain) \linebreak

This function sets the domain to search within when calls are made to gettext(), usually the named after an application. The previous default domain is returned. Call it with `NULL` as parameter to get the current setting without changing it.

XXXVI. Funzioni GMP

Queste funzioni permettono di lavorare con numeri interi di lunghezza arbitraria usando le librerie GNU MP. In pratica per poter usufruire di queste funzioni, bisogna installare il supporto GMP usando la seguente opzione `--with-gmp`.

Puoi scaricare la libreria GMP dal sito http://www.swox.com/gmp/. Dove è possibile anche scaricare il manuale GMP.

Per usare queste funzioni è necessaria la versione 2 o superiore delle librerie GMP.

Queste funzioni sono state aggiunte in PHP 4.0.4.

> **Nota:** Molte funzioni accettano argomenti numerici GMP, definiti come `risorse`più in basso. Comunque, molte di queste funzioni accetteranno anche normali argomenti numerici e stringhe, considerato ciò è quindi possibile convertire queste ultime in numero. Inoltre, se c'è una funzione che può operare velocemente su argomenti interi, questa potrebbe essere usata al posto della più lenta quando l'argomento fornito è un intero. Questo è fatto con chiarezza, così la logica vuole che tu possa utilizzare numeri interi in ogni funzione che richieda un numero GMP. Vedere anche la funzione gmp_init().

> **Attenzione**
>
> Se desideri specificare un "large integer" come costante, scrivilo tra virgolette come stringa. Se non lo fai, PHP interpreterà l'"integer literal" immediatamente, con una possibile perdita di precisione, ancora prima che la libreria `GMP` venga richiamata.

Esempio 1. Funzione fattoriale usando GMP

```
<?php
function fact ($x) {
    if ($x <= 1)
        return 1;
    else
        return gmp_mul ($x, fact ($x-1));
}

print gmp_strval (fact (1000)) . "\n";
?>
```

Questo calcolerà il fattoriale di 1000 (numero abbastanza grande) molto velocemente.

gmp_abs (PHP 4 >= 4.0.4)

Valore assoluto

resource **gmp_abs** (resource a) \linebreak

Restituisce il valore assoluto di *a*.

gmp_add (PHP 4 >= 4.0.4)

Somma di numeri

resource **gmp_add** (resource a, resource b) \linebreak

Somma due numeri GMP. Il risultato, sarà un numero GMP che rappresenta la somma degli argomenti.

gmp_and (PHP 4 >= 4.0.4)

AND logico

resource **gmp_and** (resource a, resource b) \linebreak

Calcola l'AND logico di due numeri GMP.

gmp_clrbit (PHP 4 >= 4.0.4)

Pulisce bit

resource **gmp_clrbit** (resource &a, int index) \linebreak

Ripulisce (imposta a 0) il bit *index* in *a*.

gmp_cmp (PHP 4 >= 4.0.4)

Confronto di numeri

int **gmp_cmp** (resource a, resource b) \linebreak

Restituisce un valore positivo se `a > b`, zero se `a = b` e negativo se `a < b`.

gmp_com (PHP 4 >= 4.0.4)

Calcola il complemento a uno di ’a’

resource **gmp_com** (resource a) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

gmp_div (PHP 4 >= 4.0.4)

Divisione di numberi

resource **gmp_div** (resource a, resource b) \linebreak

Questa funzione è la stessa di gmp_div_q().

gmp_div_q (PHP 4 >= 4.0.4)

Divide due numberi

resource **gmp_div_q** (resource a, resource b [, int round]) \linebreak

Divide *a* per *b* e restituisce un numero intero. L’arrotondamento del risultato, è definito dal *round*, che può avere i seguenti valori:

- *GMP_ROUND_ZERO*: Il risultato è troncato verso 0.
- *GMP_ROUND_PLUSINF*: Il risultato è arrotondato verso `+infinito`.
- *GMP_ROUND_MINUSINF*: Il risultato è arrotondato verso `-infinito`.

Questa funzione può anche essere richiamata come gmp_div().

Vedere anche gmp_div_r(), gmp_div_qr()

gmp_div_qr (PHP 4 >= 4.0.4)

Divide due numeri e restituisce quoziente e resto

array **gmp_div_qr** (resource n, resource d [, int round]) \linebreak

Questa funzione, esegue la divisione tra *n* e *d* , restituendo un array, il primo elemento è, `[n/d]` (il risultato intero della divisione) il secondo è `(n - [n/d] * d)` (cioè il resto della divisone).

Vedere la funzione gmp_div_q() per la descrizione dell'argomento *round*.

Esempio 1. Divisione di numeri GMP

```
<?php
    $a = gmp_init ("0x41682179fbf5");
    $res = gmp_div_qr ($a, "0xDEFE75");
    printf("Result is: q - %s, r - %s",
            gmp_strval ($res[0]), gmp_strval ($res[1]));
?>
```

Vedere anche gmp_div_q(), gmp_div_r().

gmp_div_r (PHP 4 >= 4.0.4)

Resto di una divisione

resource **gmp_div_r** (resource n, resource d [, int round]) \linebreak

Calcola il resto di una divisione intera fra *n* e *d*. Se non è zero, il resto ha il segno dell'argomento *n*.

Vedi la funzione gmp_div_q() per la descrizione dell'argomento *round*.

Vedere anche gmp_div_q(), gmp_div_qr()

gmp_divexact (PHP 4 >= 4.0.4)

Divisione intera di numeri

resource **gmp_divexact** (resource n, resource d) \linebreak

Divide *n* per *d*, usando un algoritmo di "esatta divisione". Questa funzione, restituisce un valore corretto solo quando è noto a priori che *d* divide *n*.

gmp_fact (PHP 4 >= 4.0.4)

Fattoriale

resource **gmp_fact** (int a) \linebreak

Calcola il fattoriale (`a!`) di *a*.

gmp_gcd (PHP 4 >= 4.0.4)

Calcola il MCD

resource **gmp_gcd** (resource a, resource b) \linebreak

Calcola il massimo comune divisore (MCD) di *a* e *b*. Il risultato è sempre positivo, anche se uno o entrambi gli operatori sono negativi.

gmp_gcdext (PHP 4 >= 4.0.4)

Calcola il MCD e moltiplicatori

array **gmp_gcdext** (resource a, resource b) \linebreak

Calcola g, s e t, in questo modo `a*s + b*t = g = gcd(a,b)`, dove MCD è il massimo comune divisore. Restituisce un array con i rispettivi argomenti, cioè, g, s e t.

gmp_hamdist (PHP 4 >= 4.0.4)

Distanza dell'hamming

int **gmp_hamdist** (resource a, resource b) \linebreak

Restituisce la distanza di hamming tra *a* e *b*. Entrambe gli operandi dovrebbero essere non-negativi.

gmp_init (PHP 4 >= 4.0.4)

Crea un numero GMP

resource **gmp_init** (mixed number) \linebreak

Crea un numero GMP partendo da un intero o da una stringa. La stringa può essere decimale o esadecimale. Nell'ultimo caso, la stringa dovrebbe iniziare con `0x`.

Esempio 1. Creare un numero GMP

```
<?php
    $a = gmp_init (123456);
    $b = gmp_init ("0xFFFFDEBACDFEDF7200");
?>
```

Nota: Non è necessario chiamare uesta funzione se si vogliono usare interi o stringhe al posto di numeri GMP nelle funzioni GMP, come gmp_add(). Se se questa conversione è possibile e necessaria, gli argomenti delle funzioni vengono automaticamente convertiti in numeri GMP, usando le stesse regole di **gmp_init()**.

gmp_intval (PHP 4 >= 4.0.4)

Converte un numero GMP in un intero

int **gmp_intval** (resource gmpnumber) \linebreak

Questa funzione converte un numero GMP in un intero.

> **Attenzione**
>
> Questa funzione restituisce un risultato utile, solo se il numero attualmente fornito al PHP è un intero (per esempio, un tipo signed long). Se desideri solo stampare il numero GMP, usa gmp_strval().

gmp_invert (PHP 4 >= 4.0.4)

Inversione di modulo

resource **gmp_invert** (resource a, resource b) \linebreak

Calcola l'inverso di *a* modulo *b*. Restituisce un valore FALSE se l'inversione non esiste.

gmp_jacobi (PHP 4 >= 4.0.4)

Simbolo di Jacobi

int **gmp_jacobi** (resource a, resource p) \linebreak

Calcola il simbolo di Jacobi (http://www.utm.edu/research/primes/glossary/JacobiSymbol.html) di *a* e *p*. *p* p potrebbe essere dispari e deve essere positivo.

gmp_legendre (PHP 4 >= 4.0.4)

Simbolo di Legendre

int **gmp_legendre** (resource a, resource p) \linebreak

Calcolo del Legendre symbol (http://www.utm.edu/research/primes/glossary/LegendreSymbol.html) di *a* e *p*. *p* potrebbe essere dispari e deve essere positivo.

gmp_mod (PHP 4 >= 4.0.4)

Modulo

resource **gmp_mod** (resource n, resource d) \linebreak

Calcola il modulo di *n* rispetto a *d*. Il risultato è sempre non-negativo, il segno di *d* viene ignorato.

gmp_mul (PHP 4 >= 4.0.4)

Prodotto di numeri

resource **gmp_mul** (resource a, resource b) \linebreak

Moltiplica *a* per *b* e restituisce il risultato.

gmp_neg (PHP 4 >= 4.0.4)

Rende un numero negativo

resource **gmp_neg** (resource a) \linebreak

Restituisce *a*.

gmp_or (PHP 4 >= 4.0.4)

OR logico

resource **gmp_or** (resource a, resource b) \linebreak

Calcola l'OR logico di due numeri GMP.

gmp_perfect_square (PHP 4 >= 4.0.4)

Controllo quadrato perfetto

bool **gmp_perfect_square** (resource a) \linebreak

Restituisce un valore vero `TRUE` se *a* è un quadrato perfetto,falso `FALSE` se non lo è.

Vedere anche: gmp_sqrt(), **gmp_sqrtrm()**.

gmp_popcount (PHP 4 >= 4.0.4)

Conteggio della popolazione

int **gmp_popcount** (resource a) \linebreak

Restituisce il conteggio della popolazione di *a*.

gmp_pow (PHP 4 >= 4.0.4)

Eleva un numero a potenza

resource **gmp_pow** (resource base, int exp) \linebreak

Eleva la base *base* ad una potenza *exp*. Nel caso di 0^0 il risultato sarà 1. L'argomento *exp* non può essere negativo.

gmp_powm (PHP 4 >= 4.0.4)

Modulo di un elevamento a potenza.

resource **gmp_powm** (resource base, resource exp, resource mod) \linebreak

Calcola il (*base* elevato a potenza *exp*) modulo *mod*. Se *exp* è negativo, il risultato sarà indefinito.

gmp_prob_prime (PHP 4 >= 4.0.4)

Controlla se il numero è "probabilmente primo"

int **gmp_prob_prime** (resource a [, int reps]) \linebreak

Se questa funzione da come risultato 0, *a* non è primo. Se sarà 1, allora *a* è "probabilmente" primo. Invece se il risultato è 2, allora *a* sarà sicuramente primo. I valori "attendibili" di *reps* possono variare da 5 a 10 (di default 10); un valore più alto fa diminuire la probabilità che un numero non primo passi come "probabile" primo.

La funzione usa il test probabilistico di Miller-Rabin.

gmp_random (PHP 4 >= 4.0.4)

Generatore di numeri casuali

resource **gmp_random** (int limiter) \linebreak

Genera un numero casuale. Il numero, sarà lungo un numero di WORD (2 byte) non superiore all'argomento *limiter*. Se l'argomento *limiter* è negativo, il numero generato sarà anch'esso negativo.

gmp_scan0 (PHP 4 >= 4.0.4)

Ricerca per 0

int **gmp_scan0** (resource a, int start) \linebreak

Cerca in *a*, partendo dal bit *start*verso i bit più significativi, fermandosi sul primo bit nullo, di cui restituisce l'indice.

gmp_scan1 (PHP 4 >= 4.0.4)

Ricerca per 1

int **gmp_scan1** (resource a, int start) \linebreak

Cerca in *a*, partendo dal bit *start*, verso i bit più significativi, fermandosi sul primo bit nullo, di cui restituisce l'indice.

gmp_setbit (PHP 4 >= 4.0.4)

Imposta bit

resource **gmp_setbit** (resource &a, int index [, bool set_clear]) \linebreak

Sets bit *index* in *a*. *set_clear* definisce se il bit è settato su 0 o su 1. Di default il bit è settato a 1.

gmp_sign (PHP 4 >= 4.0.4)

Segno di un numero

int **gmp_sign** (resource a) \linebreak

Restituisce il segno di *a* : 1 se *a* è positivo e -1 se è negativo.

gmp_sqrt (PHP 4 >= 4.0.4)

Radice quadrata

resource **gmp_sqrt** (resource a) \linebreak

Calcola la radice quadrata di *a*.

gmp_sqrtrm (unknown)

Radice quadrata con resto

array **gmp_sqrtrm** (resource a) \linebreak

Restituisce un array con due valori, il primo è la radice quadrata (fornita come intero) di *a* (vedere anche gmp_sqrt()), e il secondo è il resto (cioè, la differenza tra *a* e il primo elemento al quadrato).

gmp_strval (PHP 4 >= 4.0.4)

Converte un numero GMP in una stringa

string **gmp_strval** (resource gmpnumber [, int base]) \linebreak

Converte un numero GMP in una stringa rappresentato in base *base*. La base di default è 10. Le basi consentite variano dal 2 al 36.

Esempio 1. Converte un numero GMP in una stringa

```
<?php
    $a = gmp_init("0x41682179fbf5");
    printf ("Decimal: %s, 36-based: %s", gmp_strval($a), gmp_strval($a,36));
?>
```

gmp_sub (PHP 4 >= 4.0.4)

Sottrazione di numeri

resource **gmp_sub** (resource a, resource b) \linebreak

Sottrae *b* da *a* e restituisce il risultato.

gmp_xor (PHP 4 >= 4.0.4)

XOR logico

resource **gmp_xor** (resource a, resource b) \linebreak

Calcola l’OR logico esclusivo (XOR) di due numeri GMP.

XXXVII. Funzioni HTTP

Queste funzioni permettono di modificare l'output inviato verso un browser attraverso manipolazioni a livello di protocollo HTTP.

header (PHP 3, PHP 4 >= 4.0.0)

Spedisce un header HTTP

int **header** (string string [, bool replace]) \linebreak

header() si utilizza per inviare header HTTP. Per maggiorni informazioni riguardanti gli header HTTP si veda la risorsa HTTP 1.1 specification (http://www.w3.org/Protocols/rfc2616/rfc2616).

L'argomento opzionale *`replace`* indica se l'header inviato deve sostituirne uno simile spedito precedentemente, o accodarsi al primo dello stesso tipo. Per default la funzione sostituisce l'header precedente, ma se viene passato `FALSE` come secondo argomento vengono forzate intestazioni multiple. Per esempio:

```
header('WWW-Authenticate: Negotiate');
header('WWW-Authenticate: NTLM',false);
```

Ci sono due casi speciali di chiamate di header. Il primo è "Location". Location non trasmette solo un header al browser, ma anche un `REDIRECT` con codice di stato (302).

```
header("Location: http://www.php.net/"); /* Ridireziona il browser
                                            al sito di PHP */
exit;                 /* Assicura che il codice sottostante
                         non sia eseguito dopo il redirezionamento. */
```

> **Nota:** HTTP/1.1 richiede un URI assoluto come argomento di Location: (http://www.w3.org/Protocols/rfc2616/rfc2616-sec14.html#sec14.30) composto da schema, hostname, e path assoluto, ma alcuni clients possono accettare anche URIs relativi. E' possibile usare $HTTP_SERVER_VARS['HTTP_HOST'], $HTTP_SERVER_VARS['PHP_SELF'] e dirname() per creare URI assoluti da URI relativi in modo automatico:

```
header ("Location: http://".$HTTP_SERVER_VARS['HTTP_HOST']
                      ."/".dirname($HTTP_SERVER_VARS['PHP_SELF'])
                      ."/".$relative_url);
```

Il secondo caso speciale è esemplificato dalle intestazioni che iniziano con la stringa, "`HTTP/`" (le maiuscole non sono discriminanti), che è usato per inviare codici di stato HTTP. Per esempio, se si è configurato Apache per usare script PHP per la manipolazione di richieste fallite (usando la direttiva `ErrorDocument`), potete desiderare di assicurarvi che il vostro script generi il codice adeguato.

```
header ("HTTP/1.0 404 Not Found");
```

> **Nota:** In PHP 3, questo funziona solo se PHP è compilato come modulo Apache. Potete ottenere lo stesso effetto usando l'header `Status`.
>
> ```
> header("Status: 404 Not Found");
> ```

Spesso gli scrit PHP generano contenuti dinamici, se volete evitare che i contenuti vengano mantenuti nella cache di browser o proxy, potete forzare il loro comportamento con questa direttiva:

```
header("Expires: Mon, 26 Jul 1997 05:00:00 GMT");    // Data passata
header("Last-Modified: " . gmdate("D, d M Y H:i:s") . " GMT");
                                                     // sempre modificato
header("Cache-Control: no-store, no-cache, must-revalidate");  // HTTP/1.1
header("Cache-Control: post-check=0, pre-check=0", false);
header("Pragma: no-cache");                          // HTTP/1.0
```

> **Nota:** E' possibile che alcune pagine rimangano in chache anche dopo l'uso degli header descritti sopra. Ci sono delle opzioni che l'utente può settare dal browser, capaci di modificare i comportamenti di default del caching. Per trasmettere efficacemente gli header descritti, bisogna che sia inattiva ogni regolazione che può forzare comportamenti contrari.
>
> Inoltre, session_cache_limiter() e e la configurazione `session.cache_limiter` possono essere usate per generare automaticamente i corretti header relativi al caching durante l'uso delle sessioni.

Bisogna ricordare che la funzione**header()** va chiamata prima di qualsiasi output HTML o PHP (anche righe o spazi vuoti). E' un errore comune leggere files con funzioni include(), o require() (o altre funzioni capaci di accedere a files), che possano emettere in output spazi o linee vuote prima di una chiamata della funzione **header()**. Lo stesso problema esiste nell'utilizzare file PHP/HTML.

```
<?php require("user_logging.inc") ?>

<?php header ("Content-type: audio/x-pn-realaudio"); ?>
// Non funziona, notate le linee vuote sopra
```

> **Nota:** In PHP 4, potete usare il buffering dell'output per aggirare questo problema, evitando ogni output al browser trattenedolo al server fino a che non gli si impone l'invio. Si può fare questa operazione chiamando ob_start() e ob_end_flush() nello script, o settando la direttiva di configurazione `output_buffering` nel file `php.ini` o nel file di configurazione del server.

Se desiderate che l'utente sia spinto a salvare i dati trasmessi per esempio utilizzando un file PDF, potete usare l'header Content-Disposition (http://www.ietf.org/rfc/rfc2183.txt), che vi permette di dare un nome al file e forzare il browser a mostrare la finestra di dialogo save.

```
<?php
header("Content-type: application/pdf");
header("Content-Disposition: attachment; filename=downloaded.pdf");

/* ... manda in output un file pdf ... */
```

> **Nota:** Per un bug di Microsoft Internet Explorer 4.01 qusto sistema non funziona. Non ci sono soluzioni. C'è un altro bug in Microsoft Internet Explorer 5.5 che impedisce il giusto funzionamento, ma è possibile risolverlo con l'upgrade del Service Pack 2 o superiore.

Vedi anche headers_sent(), setcookie(), e la sezione Autenticazione HTTP usando PHP.

headers_sent (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Restituisce `TRUE` se gli header sono stati trasmessi.

bool **headers_sent** (void) \linebreak

Questa funzione restituisce `TRUE` se gli header HTTP sono stati spedite correttamente, `FALSE` in caso contrario.

Vedi anche header()

setcookie (PHP 3, PHP 4 >= 4.0.0)

Spedisce un cookie

int **setcookie** (string name [, string value [, int expire [, string path [, string domain [, int secure]]]]]) \linebreak

setcookie() definisce un cookie da inviare insieme alle altre informazioni di header. I cookie devono essere spediti *prima* di qualsiasi altra intestazione (questa è una restrizione dei cookies, non di PHP).

E’ necessario perciò chiamare la funzione **setcookie()** *prima* di qualsiasi tags, anche `<html>` o `<head>`.

Tutti gli argomenti della funzione eccetto *`name`* sono opzionali. Se viene passato alla funzione solo l’argomento *`name`*, il cookie registrato con quel nome verrà cancellato dal client su cui è archiviato. E’ possibile sostituire gli argomenti che non si intende specificare utitlizzando una stringa vuota (*""*). Gli argomenti *`expire`* e *`secure`* che richiedono numeri interi, non possono essere omessi inserendo una stringa vuota, per questo scopo si usa (*0*). L’argomento *`expire`* è un normale intero Unix Timestamp ottenibile grazie alle funzioni time() o mktime(). *`secure`* sta ad indicare che il cookie dovrebbe essere trasmesso soltanto attraverso un collegamento sicuro di tipo HTTPS.

Errori comuni:

- I cookie diventano disponibili soltanto dalla pagina successiva a quella che li ha generati, o dopo il ricaricamento di questa.
- I cookie devono essere cancellati specificando gli stessi parametri con cui sono stati creati.

In PHP 3, chiamate successive di **setcookie()** nello stesso script sono eseguite in ordine inverso. Se state provando a cancellare un cookie prima dell’ inserimento di un altro cookie, dovete creare il secondo prima della cancellazione del primo. In PHP 4, chiamate successive di **setcookie()** invece, sono eseguite secondo l’ordine di chiamata.

Alcuni esempi sul modo di spedire cookie:

Esempio 1. setcookie() esempi di spedizione/creazione

```
setcookie ("TestCookie", "Test Value");
setcookie ("TestCookie", $value,time()+3600);  /* aspira in 1 ora */
setcookie ("TestCookie", $value,time()+3600, "/~rasmus/", ".utoronto.ca", 1);
```

Gli esempi mostrano come cancellare i cookie introdotti nell’esempio precedente:

Esempio 2. setcookie() esempi di cancellazione

```
setcookie ("TestCookie");
// set the expiration date to one hour ago
setcookie ("TestCookie", "", time() - 3600);
setcookie ("TestCookie", "", time() - 3600, "/~rasmus/", ".utoronto.ca", 1);
```

Per cancellare un cookie dovete assicurarvi che la data di scadenza del cookie sia già trascorsa, in questo modo il cookie verrà rimosso dal client.

Si noti che i valori salvati nei cookies sono automaticamente codificati per la trasmissione via URL (urlencoded) quando il cookie viene inviato, e che al momento del richiamo sono automaticamente decodificati e assegnati ad una variabile che ha lo stesso nome del cookie. Per vedere il contenuto di un cookie in uno script, si usa una di queste due sintassi:

```
echo $TestCookie;
echo $HTTP_COOKIE_VARS["TestCookie"];
```

Potete registrare array in un cookie usando la notazione degli array al posto del nome del cookie. Questo equivale alla spedizione di tanti cookie quanti sono gli elementi dell'array, ma si ha un vantaggio: quando il cookie è ricevuto, tutti i suoi valori sono ordinati in un singolo array che ha per nome il nome del cookie:

```
setcookie ("cookie[three]", "cookiethree");
setcookie ("cookie[two]", "cookietwo");
setcookie ("cookie[one]", "cookieone");
if (isset ($cookie)) {
    while (list ($name, $value) = each ($cookie)) {
        echo "$name == $value&lt;br>\n";
    }
}
```

Per saperne di più sui cookies, Netscape's cookie specification è la risorsa giusta http://www.netscape.com/newsref/std/cookie_spec.html.

Microsoft Internet Explorer 4 con Service Pack 1 non crea correttamente cookie che hanno il parametro *`path`* specificato.

Netscape Communicator 4.05 e Microsoft Internet Explorer 3.x sembrano utilizzare in modo errato i cookie quando *`path`* ed *`expire`* non sono specificati.

XXXVIII. Hyperwave functions

Introduction

Hyperwave has been developed at IICM (http://www.iicm.edu/) in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (If I remember properly it was in 1996).

Hyperwave is not free software. The current version, 4.1, is available at www.hyperwave.com (http://www.hyperwave.com/). A time limited version can be ordered for free (30 days).

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user. An attribute is a name/value pair of the form name=value. The complete object record contains as many of those pairs as the user likes. The name of an attribute does not have to be unique, e.g. a title may appear several times within an object record. This makes sense if you want to specify a title in several languages. In such a case there is a convention, that each title value is preceded by the two letter language abbreviation followed by a colon, e.g. ’en:Title in English’ or ’ge:Titel in deutsch’. Other attributes like a description or keywords are potential candidates. You may also replace the language abbreviation by any other string as long as it separated by colon from the rest of the attribute value.

Each object record has native a string representation with each name/value pair separated by a newline. The Hyperwave extension also knows a second representation which is an associated array with the attribute name being the key. Multilingual attribute values itself form another associated array with the key being the language abbreviation. Actually any multiple attribute forms an associated array with the string left to the colon in the attribute value being the key. (This is not fully implemented. Only the attributes Title, Description and Keyword are treated properly yet.)

Besides the documents, all hyper links contained in a document are stored as object records as well. Hyper links which are in a document will be removed from it and stored as individual objects, when the document is inserted into the database. The object record of the link contains information about where it starts and where it ends. In order to gain the original document you will have to retrieve the plain document without the links and the list of links and reinsert them (The functions hw_pipedocument() and hw_gettext() do this for you. The advantage of separating links from the document is obvious. Once a document to which a link is pointing to changes its name, the link can easily be modified accordingly. The document containing the link is not affected at all. You may even add a link to a document without modifying the document itself.

Saying that hw_pipedocument() and hw_gettext() do the link insertion automatically is not as simple as it sounds. Inserting links implies a certain hierarchy of the documents. On a web server this is given by the file system, but Hyperwave has its own hierarchy and names do not reflect the position of an object in that hierarchy. Therefore creation of links first of all requires a mapping from the Hyperwave hierarchy and namespace into a web hierarchy respective web namespace. The fundamental difference between Hyperwave and the web is the clear distinction between names and hierarchy in Hyperwave. The name does not contain any information about the objects position in the hierarchy. In the web the name also contains the information on where the object is located in the hierarchy. This leads to two possibles ways of mapping. Either the Hyperwave hierarchy and name of the Hyperwave object is reflected in the URL or the name only. To make things simple the second approach is used. Hyperwave object with name ’my_object’ is mapped to ’http://host/my_object’

disregarding where it resides in the Hyperwave hierarchy. An object with name ’parent/my_object’ could be the child of ’my_object’ in the Hyperwave hierarchy, though in a web namespace it appears to be just the opposite and the user might get confused. This can only be prevented by selecting reasonable object names.

Having made this decision a second problem arises. How do you involve PHP? The URL http://host/my_object will not call any PHP script unless you tell your web server to rewrite it to e.g. ’http://host/php3_script/my_object’ and the script ’php3_script’ evaluates the $PATH_INFO variable and retrieves the object with name ’my_object’ from the Hyperwave server. Their is just one little drawback which can be fixed easily. Rewriting any URL would not allow any access to other document on the web server. A PHP script for searching in the Hyperwave server would be impossible. Therefore you will need at least a second rewriting rule to exclude certain URLS like all e.g. starting with http://host/Hyperwave. This is basically sharing of a namespace by the web and Hyperwave server.

Based on the above mechanism links are insert into documents.

It gets more complicated if PHP is not run as a server module or CGI script but as a standalone application e.g. to dump the content of the Hyperwave server on a CD-ROM. In such a case it makes sense to retain the Hyperwave hierarchy and map in onto the file system. This conflicts with the object names if they reflect its own hierarchy (e.g. by choosing names including ’/’). Therefore ’/’ has to be replaced by another character, e.g. ’_’. to be continued.

The network protocol to communicate with the Hyperwave server is called HG-CSP (http://www.hyperwave.com/7.17-hg-prot) (Hyper-G Client/Server Protocol). It is based on messages to initiate certain actions, e.g. get object record. In early versions of the Hyperwave Server two native clients (Harmony, Amadeus) were provided for communication with the server. Those two disappeared when Hyperwave was commercialised. As a replacement a so called wavemaster was provided. The wavemaster is like a protocol converter from HTTP to HG-CSP. The idea is to do all the administration of the database and visualisation of documents by a web interface. The wavemaster implements a set of placeholders for certain actions to customise the interface. This set of placeholders is called the PLACE Language. PLACE lacks a lot of features of a real programming language and any extension to it only enlarges the list of placeholders. This has led to the use of JavaScript which IMO does not make life easier.

Adding Hyperwave support to PHP should fill in the gap of a missing programming language for interface customisation. It implements all the messages as defined by the HG-CSP but also provides more powerful commands to e.g. retrieve complete documents.

Hyperwave has its own terminology to name certain pieces of information. This has widely been taken over and extended. Almost all functions operate on one of the following data types.

- object ID: An unique integer value for each object in the Hyperwave server. It is also one of the attributes of the object record (ObjectID). Object ids are often used as an input parameter to specify an object.
- object record: A string with attribute-value pairs of the form attribute=value. The pairs are separated by a carriage return from each other. An object record can easily be converted into an object array with **hw_object2array()**. Several functions return object records. The names of those functions end with obj.
- object array: An associated array with all attributes of an object. The key is the attribute name. If an attribute occurs more than once in an object record it will result in another indexed or associated array. Attributes which are language depended (like the title, keyword, description) will form an associated array with the key set to the language abbreviation. All other multiple

attributes will form an indexed array. PHP functions never return object arrays.

- hw_document: This is a complete new data type which holds the actual document, e.g. HTML, PDF etc. It is somewhat optimised for HTML documents but may be used for any format.

Several functions which return an array of object records do also return an associated array with statistical information about them. The array is the last element of the object record array. The statistical array contains the following entries:

Hidden

Number of object records with attribute PresentationHints set to Hidden.

CollectionHead

Number of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHead

Number of object records with attribute PresentationHints set to FullCollectionHead.

CollectionHeadNr

Index in array of object records with attribute PresentationHints set to CollectionHead.

FullCollectionHeadNr

Index in array of object records with attribute PresentationHints set to FullCollectionHead.

Total

Total: Number of object records.

Integration with Apache

The Hyperwave extension is best used when PHP is compiled as an Apache module. In such a case the underlying Hyperwave server can be hidden from users almost completely if Apache uses its rewriting engine. The following instructions will explain this.

Since PHP with Hyperwave support built into Apache is intended to replace the native Hyperwave solution based on Wavemaster I will assume that the Apache server will only serve as a Hyperwave web interface. This is not necessary but it simplifies the configuration. The concept is quite simple. First of all you need a PHP script which evaluates the PATH_INFO variable and treats its value as the name of a Hyperwave object. Let's call this script 'Hyperwave'. The URL http://your.hostname/Hyperwave/name_of_object would than return the Hyperwave object with the name 'name_of_object'. Depending on the type of the object the script has to react accordingly. If it is a collection, it will probably return a list of children. If it is a document it will return the mime type and the content. A slight improvement can be achieved if the Apache rewriting engine is used. From the users point of view it would be more straight forward if the URL http://your.hostname/name_of_object would return the object. The rewriting rule is quite easy:

```
RewriteRule ^/(.*) /usr/local/apache/htdocs/HyperWave/$1 [L]
```

Now every URL relates to an object in the Hyperwave server. This causes a simple to solve problem. There is no way to execute a different script, e.g. for searching, than the ’Hyperwave’ script. This can be fixed with another rewriting rule like the following:

```
RewriteRule ^/hw/(.*) /usr/local/apache/htdocs/hw/$1 [L]
```

This will reserve the directory `/usr/local/apache/htdocs/hw` for additional scripts and other files. Just make sure this rule is evaluated before the one above. There is just a little drawback: all Hyperwave objects whose name starts with ’hw/’ will be shadowed. So, make sure you don’t use such names. If you need more directories, e.g. for images just add more rules or place them all in one directory. Finally, don’t forget to turn on the rewriting engine with

```
RewriteEngine on
```

My experiences have shown that you will need the following scripts:

- to return the object itself
- to allow searching
- to identify yourself
- to set your profile
- one for each additional function like to show the object attributes, to show information about users, to show the status of the server, etc.

Todo

There are still some things todo:

- The hw_InsertDocument has to be split into hw_insertobject() and **hw_putdocument()**.
- The names of several functions are not fixed, yet.
- Most functions require the current connection as its first parameter. This leads to a lot of typing, which is quite often not necessary if there is just one open connection. A default connection will improve this.
- Conversion form object record into object array needs to handle any multiple attribute.

hw_Array2Objrec (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

convert attributes from object array to object record

strin **hw_array2objrec** (array object_array) \linebreak

Converts an `object_array` into an object record. Multiple attributes like 'Title' in different languages are treated properly.

See also hw_objrec2array().

hw_changeobject (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Changes attributes of an object (obsolete)

void **hw_changeobject** (int link, int objid, array attributes) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

hw_Children (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of children

array **hw_children** (int connection, int objectID) \linebreak

Returns an array of object ids. Each id belongs to a child of the collection with ID `objectID`. The array contains all children both documents and collections.

hw_ChildrenObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of children

array **hw_childrenobj** (int connection, int objectID) \linebreak

Returns an array of object records. Each object record belongs to a child of the collection with ID `objectID`. The array contains all children both documents and collections.

hw_Close (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

closes the Hyperwave connection

int **hw_close** (int connection) \linebreak

Returns FALSE if connection is not a valid connection index, otherwise TRUE. Closes down the connection to a Hyperwave server with the given connection index.

hw_Connect (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

opens a connection

int **hw_connect** (string host, int port, string username, string password) \linebreak

Opens a connection to a Hyperwave server and returns a connection index on success, or FALSE if the connection could not be made. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple connections open at once. Keep in mind, that the password is not encrypted.

See also hw_pconnect().

hw_connection_info (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Prints information about the connection to Hyperwave server

void **hw_connection_info** (int link) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

hw_Cp (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

copies objects

int **hw_cp** (int connection, array object_id_array, int destination id) \linebreak

Copies the objects with object ids as specified in the second parameter to the collection with the id *destination id*.

The value return is the number of copied objects.

See also hw_mv().

hw_Deleteobject (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

deletes object

int **hw_deleteobject** (int connection, int object_to_delete) \linebreak

Deletes the object with the given object id in the second parameter. It will delete all instances of the object.

Returns TRUE if no error occurs otherwise FALSE.

See also hw_mv().

hw_DocByAnchor (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object id object belonging to anchor

int **hw_docbyanchor** (int connection, int anchorID) \linebreak

Returns an th object id of the document to which *anchorID* belongs.

hw_DocByAnchorObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object record object belonging to anchor

string **hw_docbyanchorobj** (int connection, int anchorID) \linebreak

Returns an th object record of the document to which *anchorID* belongs.

hw_Document_Attributes (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object record of hw_document

string **hw_document_attributes** (int hw_document) \linebreak

Returns the object record of the document.

For backward compatibility, **hw_documentattributes()** is also accepted. This is deprecated, however.

See also hw_document_bodytag(), and hw_document_size().

hw_Document_BodyTag (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

body tag of hw_document

string **hw_document_bodytag** (int hw_document) \linebreak

Returns the BODY tag of the document. If the document is an HTML document the BODY tag should be printed before the document.

See also hw_document_attributes(), and hw_document_size().

For backward compatibility, **hw_documentbodytag()** is also accepted. This is deprecated, however.

hw_Document_Content (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

returns content of hw_document

string **hw_document_content** (int hw_document) \linebreak

Returns the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record.

See also hw_document_attributes(), hw_document_size(), and hw_document_setcontent().

hw_Document_SetContent (PHP 4 >= 4.0.0)

sets/replaces content of hw_document

string **hw_document_setcontent** (int hw_document, string content) \linebreak

Sets or replaces the content of the document. If the document is an HTML document the content is everything after the BODY tag. Information from the HEAD and BODY tag is in the stored in the object record. If you provide this information in the content of the document too, the Hyperwave server will change the object record accordingly when the document is inserted. Probably not a very good idea. If this functions fails the document will retain its old content.

See also hw_document_attributes(), hw_document_size(), and hw_document_content().

hw_Document_Size (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

size of hw_document

int **hw_document_size** (int hw_document) \linebreak

Returns the size in bytes of the document.

See also hw_document_bodytag(), and hw_document_attributes().

For backward compatibility, **hw_documentsize()** is also accepted. This is deprecated, however.

hw_dummy (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Hyperwave dummy function

string **hw_dummy** (int link, int id, int msgid) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

hw_EditText (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

retrieve text document

int **hw_edittext** (int connection, int hw_document) \linebreak

Uploads the text document to the server. The object record of the document may not be modified while the document is edited. This function will only works for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also hw_pipedocument(), hw_free_document(), hw_document_bodytag(), hw_document_size(), hw_output_document(), hw_gettext().

hw_Error (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

error number

int **hw_error** (int connection) \linebreak

Returns the last error number. If the return value is 0 no error has occurred. The error relates to the last command.

hw_ErrorMsg (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

returns error message

string **hw_errormsg** (int connection) \linebreak

Returns a string containing the last error message or 'No Error'. If FALSE is returned, this function failed. The message relates to the last command.

hw_Free_Document (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

frees hw_document

int **hw_free_document** (int hw_document) \linebreak

Frees the memory occupied by the Hyperwave document.

hw_GetAnchors (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of anchors of document

array **hw_getanchors** (int connection, int objectID) \linebreak

Returns an array of object ids with anchors of the document with object ID *objectID*.

hw_GetAnchorsObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of anchors of document

array **hw_getanchorsobj** (int connection, int objectID) \linebreak

Returns an array of object records with anchors of the document with object ID *objectID*.

hw_GetAndLock (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

return bject record and lock object

string **hw_getandlock** (int connection, int objectID) \linebreak

Returns the object record for the object with ID *objectID*. It will also lock the object, so other users cannot access it until it is unlocked.

See also hw_unlock(), and hw_getobject().

hw_GetChildColl (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of child collections

array **hw_getchildcoll** (int connection, int objectID) \linebreak

Returns an array of object ids. Each object ID belongs to a child collection of the collection with ID *objectID*. The function will not return child documents.

See also hw_children(), and hw_getchilddoccoll().

hw_GetChildCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of child collections

array **hw_getchildcollobj** (int connection, int objectID) \linebreak

Returns an array of object records. Each object records belongs to a child collection of the collection with ID *`objectID`*. The function will not return child documents.

See also hw_childrenobj(), and hw_getchilddoccollobj().

hw_GetChildDocColl (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of child documents of collection

array **hw_getchilddoccoll** (int connection, int objectID) \linebreak

Returns array of object ids for child documents of a collection.

See also hw_children(), and hw_getchildcoll().

hw_GetChildDocCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of child documents of collection

array **hw_getchilddoccollobj** (int connection, int objectID) \linebreak

Returns an array of object records for child documents of a collection.

See also hw_childrenobj(), and hw_getchildcollobj().

hw_GetObject (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object record

array **hw_getobject** (int connection, [int|array] objectID, string query) \linebreak

Returns the object record for the object with ID *`objectID`* if the second parameter is an integer. If the second parameter is an array of integer the function will return an array of object records. In such a case the last parameter is also evaluated which is a query string.

The query string has the following syntax:

<expr> ::= "(" <expr> ")" |

"!" <expr> | /* NOT */

<expr> "||" <expr> | /* OR */

<expr> "&&" <expr> | /* AND */

<attribute> <operator> <value>

<attribute> ::= /* any attribute name (Title, Author, DocumentType ...) */

<operator> ::= "=" | /* equal */

"<" | /* less than (string compare) */

">" | /* greater than (string compare) */

"~" /* regular expression matching */

The query allows to further select certain objects from the list of given objects. Unlike the other query functions, this query may use not indexed attributes. How many object records are returned depends on the query and if access to the object is allowed.

See also hw_getandlock(), and hw_getobjectbyquery().

hw_GetObjectByQuery (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

search object

array **hw_getobjectbyquery** (int connection, string query, int max_hits) \linebreak

Searches for objects on the whole server and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyqueryobj().

hw_GetObjectByQueryColl (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

search object in collection

array **hw_getobjectbyquerycoll** (int connection, int objectID, string query, int max_hits) \linebreak

Searches for objects in collection with ID *objectID* and returns an array of object ids. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquerycollobj().

hw_GetObjectByQueryCollObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

search object in collection

array **hw_getobjectbyquerycollobj** (int connection, int objectID, string query, int max_hits) \linebreak

Searches for objects in collection with ID *objectID* and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquerycoll().

hw_GetObjectByQueryObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

search object

array **hw_getobjectbyqueryobj** (int connection, string query, int max_hits) \linebreak

Searches for objects on the whole server and returns an array of object records. The maximum number of matches is limited to *max_hits*. If *max_hits* is set to -1 the maximum number of matches is unlimited.

The query will only work with indexed attributes.

See also hw_getobjectbyquery().

hw_GetParents (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object ids of parents

array **hw_getparents** (int connection, int objectID) \linebreak

Returns an indexed array of object ids. Each object id belongs to a parent of the object with ID *objectID*.

hw_GetParentsObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

object records of parents

array **hw_getparentsobj** (int connection, int objectID) \linebreak

Returns an indexed array of object records plus an associated array with statistical information about the object records. The associated array is the last entry of the returned array. Each object record belongs to a parent of the object with ID *objectID*.

hw_getrellink (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Get link from source to dest relative to rootid

string **hw_getrellink** (int link, int rootid, int sourceid, int destid) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

hw_GetRemote (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Gets a remote document

int **hw_getremote** (int connection, int objectID) \linebreak

Returns a remote document. Remote documents in Hyperwave notation are documents retrieved from an external source. Common remote documents are for example external web pages or queries in a database. In order to be able to access external sources throught remote documents Hyperwave introduces the HGI (Hyperwave Gateway Interface) which is similar to the CGI. Currently, only ftp, http-servers and some databases can be accessed by the HGI. Calling **hw_getremote()** returns the document from the external source. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also hw_getremotechildren().

hw_GetRemoteChildren (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Gets children of remote document

int **hw_getremotechildren** (int connection, string object record) \linebreak

Returns the children of a remote document. Children of a remote document are remote documents itself. This makes sense if a database query has to be narrowed and is explained in Hyperwave Programmers' Guide. If the number of children is 1 the function will return the document itself formated by the Hyperwave Gateway Interface (HGI). If the number of children is greater than 1 it will return an array of object record with each maybe the input value for another call to **hw_getremotechildren()**. Those object records are virtual and do not exist in the Hyperwave server, therefore they do not have a valid object ID. How exactely such an object record looks like is up to the HGI. If you want to use this function you should be very familiar with HGIs. You should also consider to use PHP instead of Hyperwave to access external sources. Adding database support by a Hyperwave gateway should be more difficult than doing it in PHP.

See also hw_getremote().

hw_GetSrcByDestObj (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns anchors pointing at object

array **hw_getsrcbydestobj** (int connection, int objectID) \linebreak

Returns the object records of all anchors pointing to the object with ID `objectID`. The object can either be a document or an anchor of type destination.

See also hw_getanchors().

hw_GetText (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

retrieve text document

int **hw_gettext** (int connection, int objectID [, mixed rootID/prefix]) \linebreak

Returns the document with object ID *objectID*. If the document has anchors which can be inserted, they will be inserted already. The optional parameter *rootID/prefix* can be a string or an integer. If it is an integer it determines how links are inserted into the document. The default is 0 and will result in links that are constructed from the name of the link's destination object. This is useful for web applications. If a link points to an object with name 'internet_movie' the HTML link will be <A HREF="/internet_movie">. The actual location of the source and destination object in the document hierachy is disregarded. You will have to set up your web browser, to rewrite that URL to for example '/my_script.php3/internet_movie'. 'my_script.php3' will have to evaluate $PATH_INFO and retrieve the document. All links will have the prefix '/my_script.php3/'. If you do not want this you can set the optional parameter *rootID/prefix* to any prefix which is used instead. Is this case it has to be a string.

If *rootID/prefix* is an integer and unequal to 0 the link is constructed from all the names starting at the object with the id *rootID/prefix* separated by a slash relative to the current object. If for example the above document 'internet_movie' is located at 'a-b-c-internet_movie' with '-' being the seperator between hierachy levels on the Hyperwave server and the source document is located at 'a-b-d-source' the resulting HTML link would be: <A HREF="../c/internet_movie">. This is useful if you want to download the whole server content onto disk and map the document hierachy onto the file system.

This function will only work for pure text documents. It will not open a special data connection and therefore blocks the control connection during the transfer.

See also hw_pipedocument(), hw_free_document(), hw_document_bodytag(), hw_document_size(), and hw_output_document().

hw_getusername (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

name of currently logged in user

string **hw_getusername** (int connection) \linebreak

Returns the username of the connection.

hw_Identify (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

identifies as user

int **hw_identify** (string username, string password) \linebreak

Identifies as user with *username* and *password*. Identification is only valid for the current session. I do not thing this function will be needed very often. In most cases it will be easier to identify with the opening of the connection.

See also hw_connect().

hw_InCollections (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

check if object ids in collections

array **hw_incollections** (int connection, array object_id_array, array collection_id_array, int return_collections) \linebreak

Checks whether a set of objects (documents or collections) specified by the *object_id_array* is part of the collections listed in *collection_id_array*. When the fourth parameter *return_collections* is 0, the subset of object ids that is part of the collections (i.e., the documents or collections that are children of one or more collections of collection ids or their subcollections, recursively) is returned as an array. When the fourth parameter is 1, however, the set of collections that have one or more objects of this subset as children are returned as an array. This option allows a client to, e.g., highlight the part of the collection hierarchy that contains the matches of a previous query, in a graphical overview.

hw_Info (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

info about connection

string **hw_info** (int connection) \linebreak

Returns information about the current connection. The returned string has the following format: <Serverstring>, <Host>, <Port>, <Username>, <Port of Client>, <Byte swapping>

hw_InsColl (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

insert collection

int **hw_inscoll** (int connection, int objectID, array object_array) \linebreak

Inserts a new collection with attributes as in *object_array* into collection with object ID *objectID*.

hw_InsDoc (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

insert document

int **hw_insdoc** (int connection, int parentID, string object_record, string text) \linebreak

Inserts a new document with attributes as in *object_record* into collection with object ID *parentID*. This function inserts either an object record only or an object record and a pure ascii text in *text* if *text* is given. If you want to insert a general document of any kind use hw_insertdocument() instead.

See also hw_insertdocument(), and hw_inscoll().

hw_insertanchors (PHP 4 >= 4.0.4)

Inserts only anchors into text

string **hw_insertanchors** (int hwdoc, array anchorecs, array dest [, array urlprefixes]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

hw_InsertDocument (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

upload any document

int **hw_insertdocument** (int connection, int parent_id, int hw_document) \linebreak

Uploads a document into the collection with *parent_id*. The document has to be created before with hw_new_document(). Make sure that the object record of the new document contains at least the attributes: Type, DocumentType, Title and Name. Possibly you also want to set the MimeType. The functions returns the object id of the new document or FALSE.

See also hw_pipedocument().

hw_InsertObject (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

inserts an object record

int **hw_insertobject** (int connection, string object rec, string parameter) \linebreak

Inserts an object into the server. The object can be any valid hyperwave object. See the HG-CSP documentation for a detailed information on how the parameters have to be.

Note: If you want to insert an Anchor, the attribute Position has always been set either to a start/end value or to 'invisible'. Invisible positions are needed if the annotation has no correspondig link in the annotation text.

See also hw_pipedocument(), hw_insertdocument(), hw_insdoc(), and hw_inscoll().

hw_mapid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Maps global id on virtual local id

int **hw_mapid** (int connection, int server id, int object id) \linebreak

Maps a global object id on any hyperwave server, even those you did not connect to with hw_connect(), onto a virtual object id. This virtual object id can then be used as any other object id, e.g. to obtain the object record with hw_getobject(). The server id is the first part of the global object id (GOid) of the object which is actually the IP number as an integer.

Note: In order to use this function you will have to set the F_DISTRIBUTED flag, which can currently only be set at compile time in hg_comm.c. It is not set by default. Read the comment at the beginning of hg_comm.c

hw_Modifyobject (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

modifies object record

int **hw_modifyobject** (int connection, int object_to_change, array remove, array add, int mode) \linebreak

This command allows to remove, add, or modify individual attributes of an object record. The object is specified by the Object ID `object_to_change`. The first array `remove` is a list of attributes to remove. The second array `add` is a list of attributes to add. In order to modify an attribute one will have to remove the old one and add a new one. **hw_modifyobject()** will always remove the attributes before it adds attributes unless the value of the attribute to remove is not a string or array.

The last parameter determines if the modification is performed recursively. 1 means recurive modification. If some of the objects cannot be modified they will be skiped without notice. hw_error() may not indicate an error though some of the objects could not be modified.

The keys of both arrays are the attributes name. The value of each array element can either be an array, a string or anything else. If it is an array each attribute value is constructed by the key of each element plus a colon and the value of each element. If it is a string it is taken as the attribute value. An empty string will result in a complete removal of that attribute. If the value is neither a string nor an array but something else, e.g. an integer, no operation at all will be performed on the attribute. This is neccessary if you want to to add a completely new attribute not just a new value for an existing attribute. If the remove array contained an empty string for that attribute, the attribute would be tried to be removed which would fail since it doesn't exist. The following addition of a new value for that attribute would also fail. Setting the value for that attribute to e.g. 0 would not even try to remove it and the addition will work.

If you would like to change the attribute 'Name' with the current value 'books' into 'articles' you will have to create two arrays and call **hw_modifyobject()**.

Esempio 1. modifying an attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => "books");
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

In order to delete/add a name=value pair from/to the object record just pass the remove/add array and set the last/third parameter to an empty array. If the attribute is the first one with that name to add, set attribute value in the remove array to an integer.

Esempio 2. adding a completely new attribute

```
// $connect is an existing connection to the Hyperwave server
// $objid is the ID of the object to modify
$remarr = array("Name" => 0);
$addarr = array("Name" => "articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Nota: Multilingual attributes, e.g. 'Title', can be modified in two ways. Either by providing the attributes value in its native form 'language':'title' or by providing an array with elements for each language as described above. The above example would than be:

Esempio 3. modifying Title attribute

```
$remarr = array("Title" => "en:Books");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

or

Esempio 4. modifying Title attribute

```
$remarr = array("Title" => array("en" => "Books"));
$addarr = array("Title" => array("en" => "Articles", "ge"=>"Artikel"));
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

This removes the english title 'Books' and adds the english title 'Articles' and the german title 'Artikel'.

Esempio 5. removing attribute

```
$remarr = array("Title" => "");
$addarr = array("Title" => "en:Articles");
$hw_modifyobject($connect, $objid, $remarr, $addarr);
```

Nota: This will remove all attributes with the name 'Title' and adds a new 'Title' attribute. This comes in handy if you want to remove attributes recursively.

> **Nota:** If you need to delete all attributes with a certain name you will have to pass an empty string as the attribute value.

> **Nota:** Only the attributes 'Title', 'Description' and 'Keyword' will properly handle the language prefix. If those attributes don't carry a language prefix, the prefix 'xx' will be assigned.

> **Nota:** The 'Name' attribute is somewhat special. In some cases it cannot be complete removed. You will get an error message 'Change of base attribute' (not clear when this happens). Therefore you will always have to add a new Name first and than remove the old one.

> **Nota:** You may not suround this function by calls to hw_getandlock() and hw_unlock(). **hw_modifyobject()** does this internally.

Returns `TRUE` if no error occurs otherwise `FALSE`.

hw_Mv (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

moves objects

int **hw_mv** (int connection, array object id array, int source id, int destination id) \linebreak

Moves the objects with object ids as specified in the second parameter from the collection with id *`source id`* to the collection with the id *`destination id`*. If the destination id is 0 the objects will be unlinked from the source collection. If this is the last instance of that object it will be deleted. If you want to delete all instances at once, use hw_deleteobject().

The value return is the number of moved objects.

See also hw_cp(), and hw_deleteobject().

hw_New_Document (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

create new document

int **hw_new_document** (string object_record, string document_data, int document_size) \linebreak

Returns a new Hyperwave document with document data set to *`document_data`* and object record set to *`object_record`*. The length of the *`document_data`* has to passed in *`document_size`*This function does not insert the document into the Hyperwave server.

See also hw_free_document(), hw_document_size(), hw_document_bodytag(), hw_output_document(), and hw_insertdocument().

hw_Objrec2Array (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

convert attributes from object record to object array

array **hw_objrec2array** (string object_record [, array format]) \linebreak

Converts an *object_record* into an object array. The keys of the resulting array are the attributes names. Multi-value attributes like 'Title' in different languages form its own array. The keys of this array are the left part to the colon of the attribute value. This left part must be two characters long. Other multi-value attributes without a prefix form an indexed array. If the optional parameter is missing the attributes 'Title', 'Description' and 'Keyword' are treated as language attributes and the attributes 'Group', 'Parent' and 'HtmlAttr' as non-prefixed multi-value attributes. By passing an array holding the type for each attribute you can alter this behaviour. The array is an associated array with the attribute name as its key and the value being one of `HW_ATTR_LANG` or `HW_ATTR_NONE`

See also hw_array2objrec().

hw_Output_Document (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

prints hw_document

int **hw_output_document** (int hw_document) \linebreak

Prints the document without the BODY tag.

For backward compatibility, **hw_outputdocument()** is also accepted. This is deprecated, however.

hw_pConnect (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

make a persistent database connection

int **hw_pconnect** (string host, int port, string username, string password) \linebreak

Returns a connection index on success, or `FALSE` if the connection could not be made. Opens a persistent connection to a Hyperwave server. Each of the arguments should be a quoted string, except for the port number. The *username* and *password* arguments are optional and can be left out. In such a case no identification with the server will be done. It is similar to identify as user anonymous. This function returns a connection index that is needed by other Hyperwave functions. You can have multiple persistent connections open at once.

See also hw_connect().

hw_PipeDocument (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

retrieve any document

int **hw_pipedocument** (int connection, int objectID) \linebreak

Returns the Hyperwave document with object ID `objectID`. If the document has anchors which can be inserted, they will have been inserted already. The document will be transfered via a special data connection which does not block the control connection.

See also hw_gettext() for more on link insertion, hw_free_document(), hw_document_size(), hw_document_bodytag(), and hw_output_document().

hw_Root (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

root object id

int **hw_root** ( ) \linebreak

Returns the object ID of the hyperroot collection. Currently this is always 0. The child collection of the hyperroot is the root collection of the connected server.

hw_setlinkroot (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Set the id to which links are calculated

void **hw_setlinkroot** (int link, int rootid) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

hw_stat (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns status string

string **hw_stat** (int link) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

hw_Unlock (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

unlock object

int **hw_unlock** (int connection, int objectID) \linebreak

Unlocks a document, so other users regain access.

See also hw_getandlock().

hw_Who (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

List of currently logged in users

int **hw_who** (int connection) \linebreak

Returns an array of users currently logged into the Hyperwave server. Each entry in this array is an array itself containing the elements id, name, system, onSinceDate, onSinceTime, TotalTime and self. 'self' is 1 if this entry belongs to the user who initianted the request.

XXXIX. Hyperwave API functions

Introduction

Hyperwave has been developed at IICM (http://www.iicm.edu/) in Graz. It started with the name Hyper-G and changed to Hyperwave when it was commercialised (If I remember properly it was in 1996).

Hyperwave is not free software. The current version, 5.5, is available at www.hyperwave.com (http://www.hyperwave.com/). A time limited version can be ordered for free (30 days).

Hyperwave is an information system similar to a database (HIS, Hyperwave Information Server). Its focus is the storage and management of documents. A document can be any possible piece of data that may as well be stored in file. Each document is accompanied by its object record. The object record contains meta data for the document. The meta data is a list of attributes which can be extended by the user. Certain attributes are always set by the Hyperwave server, other may be modified by the user.

Requirements

Since 2001 there is a Hyperwave SDK available. It supports Java, JavaScript and C++. This PHP Extension is based on the C++ interface. In order to activate the hwapi support in PHP you will have to install the Hyperwave SDK first and configure PHP with `--with-hwapi=<dir$gt;`.

Classes

The API provided by the HW_API extension is fully object oriented. It is very similar to the C++ interface of the Hyperwave SDK. It consist of the following classes.

- `HW_API`
- `HW_API_Object`
- `HW_API_Attribute`
- `HW_API_Error`
- `HW_API_Content`
- `HW_API_Reason`

Some basic classes like `HW_API_String`, `HW_API_String_Array`, etc., which exist in the Hyperwave SDK have not been implemented since PHP has powerful replacements for them.

Each class has certain method, whose names are identical to its counterparts in the Hyperwave SDK. Passing arguments to this function differs from all the other PHP Extension but is close to the C++ API of the HW SDK. Instead of passing serval parameters they are all put into an associated array and passed as one paramter. The names of the keys are identical to those documented in the HW

SDK. The common parameters are listed below. If other parameters are required they will be documented if needed.

- `objectIdentifier` The name or id of an object, e.g. "rootcollection", "0x873A8768 0x00000002".
- `parentIdentifier` The name or id of an object which is considered to be a parent.
- `object` An instance of class HW_API_Object.
- `parameters` An instance of class HW_API_Object.
- `version` The version of an object.
- `mode` An integer value determine the way an operation is executed.
- `attributeSelector` Any array of strings, each containing a name of an attribute. This is used if you retrieve the object record and want to include certain attributes.
- `objectQuery` A query to select certain object out of a list of objects. This is used to reduce the number of objects which was delivered by a function like **hw_api->children()** or **hw_api->find()**.

Integration with Apache

The integration with Apache and possible other servers is already described in the Hyperwave Modul which has been the first extension to connect a Hyperwave Server.

hw_api_attribute (unknown)

Creates instance of class hw_api_attribute

object **attribute** ([string name [, string value]]) \linebreak

Creates a new instance of hw_api_attribute with the given name and value.

hw_api_attribute->key (unknown)

Returns key of the attribute

string **key** (void) \linebreak

Returns the name of the attribute.

See also hwapi_attribute_value().

hw_api_attribute->langdepvalue (unknown)

Returns value for a given language

string **langdepvalue** (string language) \linebreak

Returns the value in the given language of the attribute.

See also hwapi_attribute_value().

hw_api_attribute->value (unknown)

Returns value of the attribute

string **value** (void) \linebreak

Returns the value of the attribute.

See also hwapi_attribute_key(), hwapi_attribute_values().

hw_api_attribute->values (unknown)

Returns all values of the attribute

array **values** (void) \linebreak

Returns all values of the attribute as an array of strings.

See also hwapi_attribute_value().

hw_api->checkin (unknown)

Checks in an object

object **checkin** (array parameter) \linebreak

This function checks in an object or a whole hiearchie of objects. The parameters array contains the required element ’objectIdentifier’ and the optional element ’version’, ’comment’, ’mode’ and ’objectQuery’. ’version’ sets the version of the object. It consists of the major and minor version separated by a period. If the version is not set, the minor version is incremented. ’mode’ can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks in and commits the object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check in is a collection, all children will be checked in recursively if they are documents. Trying to check in a collection would result in an error.

HW_API_CHECKIN_FORCE_VERSION_CONTROL

Checks in an object even if it is not under version control.

HW_API_CHECKIN_REVERT_IF_NOT_CHANGED

Check if the new version is different from the last version. Unless this is the case the object will be checked in.

HW_API_CHECKIN_KEEP_TIME_MODIFIED

Keeps the time modified from the most recent object.

HW_API_CHECKIN_NO_AUTO_COMMIT

The object is not automatically commited on checkin.

See also hwapi_checkout().

hw_api->checkout (unknown)

Checks out an object

object **checkout** (array parameter) \linebreak

This function checks out an object or a whole hiearchie of objects. The parameters array contains the required element ’objectIdentifier’ and the optional element ’version’, ’mode’ and ’objectQuery’. ’mode’ can be one of the following values:

HW_API_CHECKIN_NORMAL

Checks out an object. The object must be a document.

HW_API_CHECKIN_RECURSIVE

If the object to check out is a collection, all children will be checked out recursively if they are documents. Trying to check out a collection would result in an error.

See also hwapi_checkin().

hw_api->children (unknown)

Returns children of an object

array **children** (array parameter) \linebreak

Retrieves the children of a collection or the attributes of a document. The children can be further filtered by specifying an object query. The parameter array contains the required elements 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

The return value is an array of objects of type `HW_API_Object` or `HW_API_Error`.

See also hwapi_parents().

hw_api->content (unknown)

Returns content of an object

object **content** (array parameter) \linebreak

This function returns the content of a document as an object of type `hw_api_content`. The parameter array contains the required elements 'objectidentifier' and the optional element 'mode'. The mode can be one of the constants HW_API_CONTENT_ALLLINKS, HW_API_CONTENT_REACHABLELINKS or HW_API_CONTENT_PLAIN. HW_API_CONTENT_ALLLINKS means to insert all anchors even if the destination is not reachable. HW_API_CONTENT_REACHABLELINKS tells **hw_api_content()** to insert only reachable links and HW_API_CONTENT_PLAIN will lead to document without any links.

hw_api_content->mimetype (unknown)

Returns mimetype

string **mimetype** (void) \linebreak

Returns the mimetype of the content.

hw_api_content->read (unknown)

Read content

string **read** (string buffer, integer len) \linebreak

Reads `len` bytes from the content into the given buffer.

hw_api->copy (unknown)

Copies physically

object **copy** (array parameter) \linebreak

This function will make a physical copy including the content if it exists and returns the new object or an error object. The parameter array contains the required elements ’objectIdentifier’ and ’destinationParentIdentifier’. The optional parameter is ’attributeSelector’‘

See also hwapi_move(), hwapi_link().

hw_api->dbstat (unknown)

Returns statistics about database server

object **dbstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_hwstat(), hwapi_ftstat().

hw_api->dcstat (unknown)

Returns statistics about document cache server

object **dcstat** (array parameter) \linebreak

See also hwapi_hwstat(), hwapi_dbstat(), hwapi_ftstat().

hw_api->dstanchors (unknown)

Returns a list of all destination anchors

object **dstanchors** (array parameter) \linebreak

Retrieves all destination anchors of an object. The parameter array contains the required element ’objectIdentifier’ and the optional elements ’attributeSelector’ and ’objectQuery’.

See also hwapi_srcanchors().

hw_api->dstofsrcanchors (unknown)

Returns destination of a source anchor

object **dstofsrcanchors** (array parameter) \linebreak

Retrieves the destination object pointed by the specified source anchors. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element 'objectIdentifier' and the optional element 'attributeSelector'.

See also hwapi_srcanchors(), hwapi_dstanchors(), hwapi_objectbyanchor().

hw_api_error->count (unknown)

Returns number of reasons

int **count** (void) \linebreak

Returns the number of error reasons.

See also hwapi_error_reason().

hw_api_error->reason (unknown)

Returns reason of error

object **reason** (void) \linebreak

Returns the first error reason.

See also hwapi_error_count().

hw_api->find (unknown)

Search for objects

array **find** (array parameter) \linebreak

This functions searches for objects either by executing a key or/and full text query. The found objects can further be filtered by an optional object query. They are sorted by their importance. The second search operation is relatively slow and its result can be limited to a certain number of hits. This allows to perform an incremental search, each returning just a subset of all found documents, starting at a given index. The parameter array contains the 'keyquery' or/and 'fulltextquery' depending on who you would like to search. Optional parameters are 'objectquery', 'scope', 'lanugages' and 'attributeselector'. In case of an incremental search the optional parameters 'startIndex', numberOfObjectsToGet' and 'exactMatchUnit' can be passed.

hw_api->ftstat (unknown)

Returns statistics about fulltext server

object **ftstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_hwstat().

hwapi_hgcsp (unknown)

Returns object of class hw_api

object **hwapi_hgcsp** (string hostname [, int port]) \linebreak

Opens a connection to the Hyperwave server on host *`hostname`*. The protocol used is HGCSP. If you do not pass a port number, 418 is used.

See also **hwapi_hwtp()**.

hw_api->hwstat (unknown)

Returns statistics about Hyperwave server

object **hwstat** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_ftstat().

hw_api->identify (unknown)

Log into Hyperwave Server

object **identify** (array parameter) \linebreak

Logs into the Hyperwave Server. The parameter array must contain the elements ’username’ und ’password’.

The return value will be an object of type `HW_API_Error` if identification failed or TRUE if it was successful.

hw_api->info (unknown)

Returns information about server configuration

object **info** (array parameter) \linebreak

See also hwapi_dcstat(), hwapi_dbstat(), hwapi_ftstat(), hwapi_hwstat().

hw_api->insert (unknown)

Inserts a new object

object **insert** (array parameter) \linebreak

Insert a new object. The object type can be user, group, document or anchor. Depending on the type other object attributes has to be set. The parameter array contains the required elements ’object’ and ’content’ (if the object is a document) and the optional parameters ’parameters’, ’mode’ and ’attributeSelector’. The ’object’ must contain all attributes of the object. ’parameters’ is an object as well holding futher attributes like the destination (attribute key is ’Parent’). ’content’ is the content of the document. ’mode’ can be a combination of the following flags:

HW_API_INSERT_NORMAL

The object in inserted into the server.

HW_API_INSERT_FORCE-VERSION-CONTROL

HW_API_INSERT_AUTOMATIC-CHECKOUT

HW_API_INSERT_PLAIN

HW_API_INSERT_KEEP_TIME_MODIFIED

HW_API_INSERT_DELAY_INDEXING

See also hwapi_replace().

hw_api->insertanchor (unknown)

Inserts a new object of type anchor

object **insertanchor** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object of type anchor and sets some of the attributes required for an anchor. The parameter array contains the required elements ’object’ and ’documentIdentifier’ and the optional elements ’destinationIdentifier’, ’parameter’, ’hint’ and ’attributeSelector’. The ’documentIdentifier’ specifies the document where the anchor shall be inserted. The target of the anchor is set in ’destinationIdentifier’ if it already exists. If the target does not exists the element ’hint’ has to be set to the name of object which is supposed to be inserted later. Once it is inserted the anchor target is resolved automatically.

See also hwapi_insertdocument(), hwapi_insertcollection(), hwapi_insert().

hw_api->insertcollection (unknown)

Inserts a new object of type collection

object **insertcollection** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object of type collection and sets some of the attributes required for a collection. The parameter array contains the required elements ’object’ and ’parentIdentifier’ and the optional elements ’parameter’ and ’attributeSelector’. See hwapi_insert() for the meaning of each element.

See also hwapi_insertdocument(), hwapi_insertanchor(), hwapi_insert().

hw_api->insertdocument (unknown)

Inserts a new object of type document

object **insertdocument** (array parameter) \linebreak

This function is a shortcut for hwapi_insert(). It inserts an object with content and sets some of the attributes required for a document. The parameter array contains the required elements ’object’, ’parentIdentifier’ and ’content’ and the optional elements ’mode’, ’parameter’ and ’attributeSelector’. See hwapi_insert() for the meaning of each element.

See also hwapi_insert() hwapi_insertanchor(), hwapi_insertcollection().

hw_api->link (unknown)

Creates a link to an object

object **link** (array parameter) \linebreak

Creates a link to an object. Accessing this link is like accessing the object to links points to. The parameter array contains the required elements ’objectIdentifier’ and ’destinationParentIdentifier’. ’destinationParentIdentifier’ is the target collection.

The function returns true on success or an error object.

See also hwapi_copy().

hw_api->lock (unknown)

Locks an object

object **lock** (array parameter) \linebreak

Locks an object for exclusive editing by the user calling this function. The object can be only unlocked by this user or the sytem user. The parameter array contains the required element ’objectIdentifier’ and the optional parameters ’mode’ and ’objectquery’. ’mode’ determines how an object is locked. HW_API_LOCK_NORMAL means, an object is locked until it is unlocked.

HW_API_LOCK_RECURSIVE is only valid for collection and locks all objects within th collection und possible subcollections. HW_API_LOCK_SESSION means, an object is locked only as long as the session is valid.

See also hwapi_unlock().

hw_api->move (unknown)

Moves object between collections

object **move** (array parameter) \linebreak

See also hw_objrec2array().

hw_api_content (unknown)

Create new instance of class hw_api_content

string **content** (string content, string mimetype) \linebreak

Creates a new content object from the string *`content`*. The mimetype is set to *`mimetype`*.

hw_api->object (unknown)

Retrieve attribute information

object **hw_api->object** (array parameter) \linebreak

This function retrieves the attribute information of an object of any version. It will not return the document content. The parameter array contains the required elements ’objectIdentifier’ and the optional elements ’attributeSelector’ and ’version’.

The returned object is an instance of class `HW_API_Object` on success or `HW_API_Error` if an error occured.

This simple example retrieves an object and checks for errors.

Esempio 1. Retrieve an object

```
<?php
function handle_error($error) {
  $reason = $error->reason(0);
  echo "Type: <B>";
  switch($reason->type()) {
    case 0:
      echo "Error";
      break;
    case 1:
      echo "Warning";
```

```
break;
    case 2:
      echo "Message";
      break;
  }
  echo "</B><BR>\n";
  echo "Description: ".$reason->description("en")."<BR>\n";
}

function list_attr($obj) {
  echo "<TABLE>\n";
  $count = $obj->count();
  for($i=0; $i<$count; $i++) {
    $attr = $obj->attribute($i);
    printf("  <TR><TD ALIGN=right bgcolor=#c0c0c0><B>%s</B></TD><TD bgcolor=#F0F0F0>%s</T
             $attr->key(), $attr->value());
  }
  echo "</TABLE>\n";
}

$hwapi = hwapi_hgcsp($g_config[HOSTNAME]);
$parms = array("objectIdentifier"=>"rootcollection, "attributeSelector"=>array("Title", "
$root = $hwapi->object($parms);
if(get_class($root) == "HW_API_Error") {
  handle_error($root);
  exit;
}
list_attr($root);
?>
```

See also hwapi_content().

hw_api_object->assign (unknown)

Clones object

object **assign** (array parameter) \linebreak

Clones the attributes of an object.

hw_api_object->attreditable (unknown)

Checks whether an attribute is editable

bool **attreditable** (array parameter) \linebreak

hw_api_object->count (unknown)

Returns number of attributes

int **count** (array parameter) \linebreak

hw_api_object->insert (unknown)

Inserts new attribute

bool **insert** (object attribute) \linebreak

Adds an attribute to the object. Returns true on success and otherwise false.

See also hwapi_object_remove().

hw_api_object (unknown)

Creates a new instance of class hw_api_object

object **hw_api_object** (array parameter) \linebreak

See also hwapi_lock().

hw_api_object->remove (unknown)

Removes attribute

bool **remove** (string name) \linebreak

Removes the attribute with the given name. Returns true on success and otherwise false.

See also hwapi_object_insert().

hw_api_object->title (unknown)

Returns the title attribute

string **title** (array parameter) \linebreak

hw_api_object->value (unknown)

Returns value of attribute

string **value** (string name) \linebreak

Returns the value of the attribute with the given name or false if an error occured.

hw_api->objectbyanchor (unknown)

Returns the object an anchor belongs to

object **objectbyanchor** (array parameter) \linebreak

This function retrieves an object the specified anchor belongs to. The parameter array contains the required element 'objectIdentifier' and the optional element 'attributeSelector'.

See also **hwapi_dstofsrcanchor()**, hwapi_srcanchors(), hwapi_dstanchors().

hw_api->parents (unknown)

Returns parents of an object

array **parents** (array parameter) \linebreak

Retrieves the parents of an object. The parents can be further filtered by specifying an object query. The parameter array contains the required elements 'objectidentifier' and the optional elements 'attributeselector' and 'objectquery'.

The return value is an array of objects of type `HW_API_Object` or `HW_API_Error`.

See also hwapi_children().

hw_api_reason->description (unknown)

Returns description of reason

string **description** (void) \linebreak

Returns the description of a reason

hw_api_reason->type (unknown)

Returns type of reason

object **type** (void) \linebreak

Returns the type of a reason.

hw_api->remove (unknown)

Delete an object

object **remove** (array parameter) \linebreak

This function removes an object from the specified parent. Collections will be removed recursively. You can pass an optional object query to remove only those objects which match the query. An object will be deleted physically if it is the last instance. The parameter array contains the required elements ’objectidentifier’ and ’parentidentifier’. If you want to remove a user or group ’parentidentifier’ can be skiped. The optional parameter ’mode’ determines how the deletion is performed. In normal mode the object will not be removed physically until all instances are removed. In physical mode all instances of the object will be deleted imediately. In removelinks mode all references to and from the objects will be deleted as well. In nonrecursive the deletion is not performed recursive. Removing a collection which is not empty will cause an error.

See also hwapi_move().

hw_api->replace (unknown)

Replaces an object

object **replace** (array parameter) \linebreak

Replaces the attributes and the content of an object The parameter array contains the required elements ’objectIdentifier’ and ’object’ and the optional parameters ’content’, ’parameters’, ’mode’ and ’attributeSelector’. ’objectIdentifier’ contains the object to be replaced. ’object’ contains the new object. ’content’ contains the new content. ’parameters’ contain extra information for HTML documents. HTML_Language is the letter abbreviation of the language of the title. HTML_Base sets the base attribute of the HTML document. ’mode’ can be a combination of the following flags:

HW_API_REPLACE_NORMAL

The object on the server is replace with the object passed.

HW_API_REPLACE_FORCE_VERSION_CONTROL

HW_API_REPLACE_AUTOMATIC_CHECKOUT

HW_API_REPLACE_AUTOMATIC_CHECKIN

HW_API_REPLACE_PLAIN

HW_API_REPLACE_REVERT_IF_NOT_CHANGED

HW_API_REPLACE_KEEP_TIME_MODIFIED

See also hwapi_insert().

hw_api->setcommitedversion (unknown)

Commits version other than last version

object **setcommitedversion** (array parameter) \linebreak

Commits a version of a docuemnt. The commited version is the one which is visible to users with read access. By default the last version is the commited version.

See also hwapi_checkin(), hwapi_checkout(), **hwapi_revert()**.

hw_api->srcanchors (unknown)

Returns a list of all source anchors

object **srcanchors** (array parameter) \linebreak

Retrieves all source anchors of an object. The parameter array contains the required element 'objectIdentifier' and the optional elements 'attributeSelector' and 'objectQuery'.

See also hwapi_dstanchors().

hw_api->srcsofdst (unknown)

Returns source of a destination object

object **srcsofdst** (array parameter) \linebreak

Retrieves all the source anchors pointing to the specified destination. The destination object can either be a destination anchor or a whole document. The parameters array contains the required element 'objectIdentifier' and the optional element 'attributeSelector' and 'objectQuery'. The function returns an array of objects or an error.

See also **hwapi_dstofsrcanchor()**.

hw_api->unlock (unknown)

Unlocks a locked object

object **unlock** (array parameter) \linebreak

Unlocks a locked object. Only the user who has locked the object and the system user may unlock an object. The parameter array contains the required element 'objectIdentifier' and the optional parameters 'mode' and 'objectquery'. The meaning of 'mode' is the same as in function hwapi_lock().

Returns true on success or an object of class HW_API_Error.

See also hwapi_lock().

hw_api->user (unknown)

Returns the own user object

object **user** (array parameter) \linebreak

See also hwapi_userlist().

hw_api->userlist (unknown)

Returns a list of all logged in users

object **userlist** (array parameter) \linebreak

See also hwapi_user().

XL. ICAP Functions [deprecated]

To get these functions to work, you have to compile PHP with `--with-icap`. That requires the icap library to be installed, which is not longer supported and available.

> **Nota:** Icap will be removed in near future. Neither this module, nor those versions of icap library are supported any longer. If you want to use calendar capabilities in php, use mcal instead.

icap_close (unknown)

Close an ICAP stream

int **icap_close** (int icap_stream [, int flags]) \linebreak

Closes the given icap stream.

icap_create_calendar (PHP 4 >= 4.0.0)

Create a new calendar

string **icap_create_calendar** (int stream_id, string calendar) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

icap_delete_calendar (PHP 4 >= 4.0.0)

Delete a calendar

string **icap_delete_calendar** (int stream_id, string calendar) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

icap_delete_event (PHP 4 >= 4.0.0)

Delete an event from an ICAP calendar

string **icap_delete_event** (int stream_id, int uid) \linebreak

icap_delete_event() deletes the calendar event specified by the *uid*.

Returns TRUE.

icap_fetch_event (PHP 4 >= 4.0.0)

Fetches an event from the calendar stream/

int **icap_fetch_event** (int stream_id, int event_id [, int options]) \linebreak

icap_fetch_event() fetches an event from the calendar stream specified by *event_id*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - TRUE if the event if public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

icap_list_alarms (PHP 4 >= 4.0.0)

Return a list of events that has an alarm triggered at the given datetime

int **icap_list_alarms** (int stream_id, array date, array time) \linebreak

Returns an array of event ID's that has an alarm going off at the given datetime.

icap_list_alarms() function takes in a datetime for a calendar stream. An array of event id's that has an alarm should be going off at the datetime are returned.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour

- int min - minutes
- int sec - seconds

icap_list_events (PHP 4 >= 4.0.0)

Return a list of events between two given datetimes

array **icap_list_events** (int stream_id, int begin_date [, int end_date]) \linebreak

Returns an array of event ID's that are between the two given datetimes.

icap_list_events() function takes in a beginning datetime and an end datetime for a calendar stream. An array of event id's that are between the given datetimes are returned.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

icap_open (PHP 4 >= 4.0.0)

Opens up an ICAP connection

stream **icap_open** (string calendar, string username, string password, string options) \linebreak

Returns an ICAP stream on success, `FALSE` on error.

icap_open() opens up an ICAP connection to the specified *`calendar`* store. If the optional *`options`* is specified, passes the *`options`* to that mailbox also.

icap_rename_calendar (PHP 4 >= 4.0.0)

Rename a calendar

string **icap_rename_calendar** (int stream_id, string old_name, string new_name) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

icap_reopen (PHP 4 >= 4.0.0)

Reopen ICAP stream to new calendar

int **icap_reopen** (int stream_id, string calendar [, int options]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

icap_snooze (PHP 4 >= 4.0.0)

Snooze an alarm

string **icap_snooze** (int stream_id, int uid) \linebreak

icap_snooze() turns on an alarm for a calendar event specified by the *uid*.

Returns TRUE.

icap_store_event (PHP 4 >= 4.0.0)

Store an event into an ICAP calendar

string **icap_store_event** (int stream_id, object event) \linebreak

icap_store_event() Stores an event into an ICAP calendar. An event object consists of:

- int public - 1 if public, 0 if private;
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - Number of minutes before the event to sned out an alarm.
- datetime start - datetime object of the start of the event.
- datetime end - datetime object of the end of the event.

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds

Returns `TRUE` on success and `FALSE` on error.

XLI. funzioni iconv

Qesto moudlo contiene un'interfaccia con le funzioni della libreria iconv. Per essere abilitati ad usare queste funzioni definite in questo modulo si deve compilare l'interprete PHP usando l'opzione --with-iconv. Per fare così, devi avere la funzione iconv() nella libreria standard di C o libriconv installata sul tuo sistema. La libreria libiconv è reperebile presso l'indirizzo http://www.gnu.org/software/libiconv/ (http://www.gnu.org/software/libiconv/).

La libreria iconv converte i file tra vari set di caratteri. I set di caratteri supportati dipendono dall'implementazione di iconv() sul tuo sistema. Nota che la funzione iconv() non è utilizzabile come ti aspetti su alcuni sistemi. In questo caso, dovresti installare la libreria libiconv.

iconv (PHP 4 >= 4.0.5)

Converte una stringa nel set di caratteri richiesto

string **iconv** (string in_charset, string out_charset, string str) \linebreak

Converte la stringa codificata nel parametro *`stringa`* in *`in_charset`* nella stringa codificata in *`out_charset`*. Restituisce la stringa convertita o `FALSE`, se fallisce.

Esempio 1. Esempio di iconv():

```
echo iconv("ISO-8859-1","UTF-8","This is test.");
```

iconv_get_encoding (PHP 4 >= 4.0.5)

Visualizza l'attuale impostazione per la conversione dei caratteri codificati

array **iconv_get_encoding** ([string type]) \linebreak

Restituisce l'attuale impostazione di ob_iconv_handler() come array o `FALSE` in caso di insuccesso.

Vedere anche: iconv_set_encoding() e ob_iconv_handler().

iconv_set_encoding (PHP 4 >= 4.0.5)

Setta l'attuale impostazione per la conversione dei caratteri codificati

array **iconv_set_encoding** (string type, string charset) \linebreak

Cambia il valore di *`type`* in *`charset`* e restituisc `TRUE` in caso di successo o `FALSE` in caso di fallimento.

Esempio 1. Esempio di iconv_set_encoding():

```
iconv_set_encoding("internal_encoding", "UTF-8");
iconv_set_encoding("output_encoding", "ISO-8859-1");
```

Vedere anche: iconv_get_encoding() e ob_iconv_handler().

ob_iconv_handler (PHP 4 >= 4.0.5)

Converte caratteri codificati come un output buffer handler

array **ob_iconv_handler** (string contents, int status) \linebreak

Converte la stringa codificata del parametro *`internal_encoding`* in *`output_encoding`*.

`internal_encoding` e *`output_encoding`* dovrebbero essere definite da iconv_set_encoding() o nel file di configurazione.

Esempio 1. Esempio di ob_iconv_handler():

```
ob_start("ob_iconv_handler"); // start output buffering
```

Vedere anche: iconv_get_encoding() e iconv_set_encoding().

XLII. Image functions

You can use the image functions in PHP to get the size of JPEG, GIF, PNG, SWF, TIFF and JPEG2000 images, and if you have the GD library (available at http://www.boutell.com/gd/) you will also be able to create and manipulate images.

If you have PHP compiled with `--enable-exif` you are able to work with information stored in headers of JPEG and TIFF images. These functions do not require GD library.

The format of images you are able to manipulate depend on the version of gd you install, and any other libraries gd might need to access those image formats. Versions of gd older than gd-1.6 support gif format images, and do not support png, where versions greater than gd-1.6 support png, not gif.

In order to read and write images in jpeg format, you will need to obtain and install jpeg-6b (available at ftp://ftp.uu.net/graphics/jpeg/), and then recompile gd to make use of jpeg-6b. You will also have to compile PHP with `--with-jpeg-dir=/path/to/jpeg-6b`.

To add support for Type 1 fonts, you can install t1lib (available at ftp://sunsite.unc.edu/pub/Linux/libs/graphics/), and then add `--with-t1lib[=dir]`.

exif_imagetype (PHP 4 CVS only)

Determine the type of an image

int|false **exif_imagetype** (string filename) \linebreak

exif_imagetype() reads the first bytes of an image and checks its signature. When a correct signature is found a constant will be returned otherwise the return value is `FALSE`. The return value is the same value that getimagesize() returns in index 2 but this function is much faster.

The following constants are defined: 1 = IMAGETYPE_GIF, 2 = IMAGETYPE_JPG, 3 = IMAGETYPE_PNG, 4 = IMAGETYPE_SWF, 5 = IMAGETYPE_PSD, 6 = IMAGETYPE_BMP, 7 = IMAGETYPE_TIFF_II (intel byte order), 8 = IMAGETYPE_TIFF_MM (motorola byte order), 9 = IMAGETYPE_JPC, 10 = IMAGETYPE_JP2, 11 = IMAGETYPE_JPX.

This function can be used to avoid calls to other exif functions with unsupported file teypes or in conjunction with $_SERVER['HTTP_ACCEPT'] to check whether or not the viewer is able to see a specific image in his browser.

> **Nota:** This function is only available in PHP 4 compiled using `--enable-exif`.
>
> This function does not require the GD image library.
>
> See also getimagesize().

exif_read_data (PHP 4 >= 4.2.0)

Read the EXIF headers from JPEG or TIFF

array **exif_read_data** (string filename [, string sections [, bool arrays [, bool thumbnail]]]) \linebreak

The **exif_read_data()** function reads the EXIF headers from a JPEG or TIFF image file. It returns an associative array where the indexes are the header names and the values are the values associated with those headers. If no data can be returned the result is `FALSE`.

`filename` is the name of the file to read. This cannot be a url.

`sections` a comma separated lsit of sections that need to be present in file to produce a result array.

FILE	FileName, FileSize, FileDateTime, SectionsFound
COMPUTED	html, Width, Height, IsColor and some more if available.
ANY_TAG	Any information that has a Tag e.g. IFD0, EXIF, ...
IFD0	All tagged data of IFD0. In normal imagefiles this contains image size and so forth.

THUMBNAIL	A file is supposed to contain a thumbnail if it has a second IFD. All tagged information about the embedded thumbnail is stored in this section.
COMMENT	Comment headers of JPEG images.
EXIF	The EXIF section is a sub section of IFD0. It contains more detailed information about an image. Most of these entries are digital camera related.

`arrays` specifies whether or not each section becomes an array. The sections *FILE*, *COMPUTED* and *THUMBNAIL* allways become arrays as they may contain values whose names are conflict with other sections.

`thumbnail` whether or not to read the thumbnail itself and not only its tagged data.

> **Nota:** Exif headers tend to be present in JPEG/TIFF images generated by digital cameras, but unfortunately each digital camera maker has a different idea of how to actually tag their images, so you can't always rely on a specific Exif header being present.

Esempio 1. exif_read_data() example

```
<?php
echo "test1.jpg:<br>\n";
$exif = exif_read_data ('tests/test1.jpg','IFD0');
echo $exif===false ? "No header data found.<br>\n" : "Image contains headers<br>";
$exif = exif_read_data ('tests/test2.jpg',0,true);
echo "test2.jpg:<br>\n";
foreach($exif as $key=>$section) {
    foreach($section as $name=>$val) {
        echo "$key.$name: $val<br>\n";
    }
}?>
```

The first call fails because the image has no header information.

```
test1.jpg:
No header data found.
test2.jpg:
FILE.FileName: test2.jpg
FILE.FileDateTime: 1017666176
FILE.FileSize: 1240
FILE.FileType: 2
FILE.SectionsFound: ANY_TAG, IFD0, THUMBNAIL, COMMENT
COMPUTED.html: width="1" height="1"
COMPUTED.Height: 1
COMPUTED.Width: 1
COMPUTED.IsColor: 1
COMPUTED.ByteOrderMotorola: 1
```

```
COMPUTED.UserComment: Exif test image.
COMPUTED.UserCommentEncoding: ASCII
COMPUTED.Copyright: Photo (c) M.Boerger, Edited by M.Boerger.
COMPUTED.Copyright.Photographer: Photo (c) M.Boerger
COMPUTED.Copyright.Editor: Edited by M.Boerger.
IFD0.Copyright: Photo (c) M.Boerger
IFD0.UserComment: ASCII
THUMBNAIL.JPEGInterchangeFormat: 134
THUMBNAIL.JPEGInterchangeFormatLength: 523
COMMENT.0: Comment #1.
COMMENT.1: Comment #2.
COMMENT.2: Comment #3end
```

Nota: If the image contains any IFD0 data then COMPUTED contains the entry ByteOrderMotorola which is 0 for little-endian (intel) and 1 for big-endian (motorola) byte order. This was added in PHP 4.3.

When an Exif header contains a Copyright note this itself can contain two values. As the solution is inconsitent in the Exif 2.10 standard the COMPUTED section will return both entries *Copyright.Photographer* and *Copyright.Editor* while the IFD0 sections contains the byte array with the NULL character that splits both entries. Or just the first entry if the datatype was wrong (normal behaviour of Exif). The COMPUTED will contain also an entry *Copyright* Which is either the original copyright string or it is a comma separated list of photo and editor copyright.

Nota: The tag UserComment has the same problem as the Copyright tag. It can store two values first the encoding used and second the value itself. If so the IFD section only contains the encoding or a byte array. The COMPUTED section will store both in the entries *UserCommentEncoding* and *UserComment*. The entry *UserComment* is available in both cases so it should be used in preference to the value in IFD0 section.

If the user comment uses Unicode or JIS encoding and the module mbstring is available this encoding will automatically changed according to the exif ini settings. This was added in PHP 4.3.

Nota: Height and Width are computed the same way getimagesize() does so their values must not be part of any header returned. Also html is a height/width text string to be used inside normal HTML.

Nota: Starting from PHP 4.3 the function can read all embedded IFD data including arrays (returned as such). Also the size of an embedded thumbnail is returned in *THUMBNAIL* subarray and the function **exif_read_data()** can return thumbnails in TIFF format. Last but not least there is no longer a maximum legth for returned values (not until memory limit is reached).

> **Nota:** This function is only available in PHP 4 compiled using `--enable-exif`. Its functionality and behaviour has changed in PHP 4.2. Earlier versions are very unstable.
>
> Since PHP 4.3 user comment can automatically change encoding if PHP 4 was compiled using `--enable-mbstring`.
>
> This function does not require the GD image library.
>
> See also exif_thumbnail() and getimagesize().

exif_thumbnail (PHP 4 >= 4.2.0)

Retrieve the embedded thumbnail of a TIFF or JPEG image

string **exif_thumbnail** (string filename [, int &width [, int &height]]) \linebreak

exif_thumbnail() reads the embedded thumbnail of a TIFF or JPEG image. If the image contains no thumbnail `FALSE` will be returned.

Both parameters *`width`* and *`height`* are available since PHP 4.3 and return the size of the thumbnail. It is possible that **exif_thumbnail()** cannot create an image but determine its size. In this case the return value is `FALSE` but *`width`* and *`height`* are set.

Starting from version PHP 4.3 the function **exif_thumbnail()** can return thumbnails in TIFF format.

> **Nota:** This function is only available in PHP 4 compiled using `--enable-exif`. Its functionality and behaviour has changed in PHP 4.2
>
> This function does not require the GD image library.
>
> See also exif_read_data().

getimagesize (PHP 3, PHP 4 >= 4.0.0)

Get the size of an image

array **getimagesize** (string filename [, array imageinfo]) \linebreak

The **getimagesize()** function will determine the size of any GIF, JPG, PNG, SWF, PSD, TIFF or BMP image file and return the dimensions along with the file type and a height/width text string to be used inside a normal HTML `IMG` tag.

Returns an array with 4 elements. Index 0 contains the width of the image in pixels. Index 1 contains the height. Index 2 a flag indicating the type of the image. 1 = GIF, 2 = JPG, 3 = PNG, 4 = SWF, 5 = PSD, 6 = BMP, 7 = TIFF(intel byte order), 8 = TIFF(motorola byte order, 9 = JPC, 10 = JP2, 11 = JPX. Index 3 is a text string with the correct height="yyy" width="xxx" string that can be used directly in an IMG tag.

Esempio 1. getimagesize (file)

```
<?php
$size = getimagesize ("img/flag.jpg");
echo "<img src=\"img/flag.jpg\" {$size[3]}>";
?>
```

Esempio 2. getimagesize (URL)

```
<?php $size = getimagesize ("http://www.example.com/gifs/logo.gif"); ?>
```

With JPG images, two extras index are returned : `channel` and `bits`. `channel` will be 3 for RGB pictures, and 4 for CMYK pictures. `bits` is the number of bits for each color.

If accessing the *`filename`* image is impossible, or if it isn't a valid picture, **getimagesize()** will return `NULL` and generate a warning.

The optional *`imageinfo`* parameter allows you to extract some extended information from the image file. Currently this will return the different JPG APP markers in an associative Array. Some Programs use these APP markers to embedd text information in images. A very common one in to embed IPTC http://www.iptc.org/ information in the APP13 marker. You can use the iptcparse() function to parse the binary APP13 marker into something readable.

Esempio 3. getimagesize returning IPTC

```
<?php
$size = getimagesize ("testimg.jpg",&$info);
if (isset ($info["APP13"])) {
    $iptc = iptcparse ($info["APP13"]);
    var_dump ($iptc);
}
?>
```

> **Nota:** TIFF support was added in PHP 4.2. JPEG2000 support will be added in PHP 4.3.
>
> This function does not require the GD image library.
>
> See also exif_imagetype(), exif_read_data() and exif_thumbnail().
>
> URL support was added in PHP 4.0.5

image2wbmp (PHP 4 >= 4.0.5)

Output image to browser or file

int **image2wbmp** (resource image [, string filename [, int threshold]]) \linebreak

image2wbmp() creates the WBMP file in filename from the image *image*. The *image* argument is the return from imagecreate().

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using header(), you can create a PHP script that outputs WBMP images directly.

> **Nota:** WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also imagewbmp().

imagealphablending (PHP 4 >= 4.0.6)

Set the blending mode for an image

int **imagealphablending** (resource image, bool blendmode) \linebreak

imagealphablending() allows for two different modes of drawing on truecolor images. In blending mode, the alpha channel component of the color supplied to all drawing function, such as imagesetpixel() determines how much of the underlying color should be allowed to shine through. As a result, gd automatically blends the existing color at that point with the drawing color, and stores the result in the image. The resulting pixel is opaque. In non-blending mode, the drawing color is copied literally with its alpha channel information, replacing the destination pixel. Blending mode is not available when drawing on palette images. If *blendmode* is `TRUE`, then blending mode is enabled, otherwise disabled.

> **Nota:** This function was added in PHP 4.0.6 and requires GD 2.0.1

imagearc (PHP 3, PHP 4 >= 4.0.0)

Draw a partial ellipse

int **imagearc** (resource image, int cx, int cy, int w, int h, int s, int e, int col) \linebreak

imagearc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e* arguments. 0° is located at the three-o'clock position, and the arc is drawn counter-clockwise.

See also imageellipse(), imagefilledellipse(), and imagefilledarc().

imagechar (PHP 3, PHP 4 >= 4.0.0)

Draw a character horizontally

int **imagechar** (resource image, int font, int x, int y, string c, int col) \linebreak

imagechar() draws the first character of *c* in the image identified by *id* with its upper-left at *x*,*y* (top left is 0, 0) with the color *col*. If *font* is 1, 2, 3, 4 or 5, a built-in font is used (with higher numbers corresponding to larger fonts).

See also imageloadfont().

imagecharup (PHP 3, PHP 4 >= 4.0.0)

Draw a character vertically

int **imagecharup** (resource image, int font, int x, int y, string c, int col) \linebreak

imagecharup() draws the character *c* vertically in the image identified by *image* at coordinates *x*, *y* (top left is 0, 0) with the color *col*. If *font* is 1, 2, 3, 4 or 5, a built-in font is used.

See also imageloadfont().

imagecolorallocate (PHP 3, PHP 4 >= 4.0.0)

Allocate a color for an image

int **imagecolorallocate** (resource image, int red, int green, int blue) \linebreak

imagecolorallocate() returns a color identifier representing the color composed of the given RGB components. The *im* argument is the return from the imagecreate() function. *red*, *green* and *blue* are the values of the red, green and blue component of the requested color respectively. These parameters are integers between 0 and 255. **imagecolorallocate()** must be called to create each color that is to be used in the image represented by *image*.

```
$white = imagecolorallocate ($im, 255, 255, 255);
$black = imagecolorallocate ($im, 0, 0, 0);
```

Returns -1 if the allocation failed.

imagecolorat (PHP 3, PHP 4 >= 4.0.0)

Get the index of the color of a pixel

int **imagecolorat** (resource image, int x, int y) \linebreak

Returns the index of the color of the pixel at the specified location in the image specified by `image`.

See also imagecolorset() and imagecolorsforindex().

imagecolorclosest (PHP 3, PHP 4 >= 4.0.0)

Get the index of the closest color to the specified color

int **imagecolorclosest** (resource image, int red, int green, int blue) \linebreak

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value.

The "distance" between the desired color and each color in the palette is calculated as if the RGB values represented points in three-dimensional space.

See also imagecolorexact().

imagecolorclosestalpha (PHP 4 >= 4.0.6)

Get the index of the closest color to the specified color + alpha

int **imagecolorclosestalpha** (resource image, int red, int green, int blue, int alpha) \linebreak

Returns the index of the color in the palette of the image which is "closest" to the specified RGB value and `alpha` level.

See also imagecolorexactalpha().

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1

imagecolorclosesthwb (PHP 4)

Get the index of the color which has the hue, white and blackness nearest to the given color

int **imagecolorclosesthwb** (resource image, int red, int green, int blue) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imagecolordeallocate (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

De-allocate a color for an image

int **imagecolordeallocate** (resource image, int color) \linebreak

The **imagecolordeallocate()** function de-allocates a color previously allocated with the imagecolorallocate() function.

```
$white = imagecolorallocate ($im, 255, 255, 255);
imagecolordeallocate ($im, $white);
```

imagecolorexact (PHP 3, PHP 4 >= 4.0.0)

Get the index of the specified color

int **imagecolorexact** (resource image, int red, int green, int blue) \linebreak

Returns the index of the specified color in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

See also imagecolorclosest().

imagecolorexactalpha (PHP 4 >= 4.0.6)

Get the index of the specified color + alpha

int **imagecolorexactalpha** (resource image, int red, int green, int blue, int alpha) \linebreak

Returns the index of the specified color+alpha in the palette of the image.

If the color does not exist in the image's palette, -1 is returned.

See also imagecolorclosestalpha().

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagecolorresolve (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Get the index of the specified color or its closest possible alternative

int **imagecolorresolve** (resource image, int red, int green, int blue) \linebreak

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also imagecolorclosest().

imagecolorresolvealpha (PHP 4 >= 4.0.6)

Get the index of the specified color + alpha or its closest possible alternative

int **imagecolorresolvealpha** (resource image, int red, int green, int blue, int alpha) \linebreak

This function is guaranteed to return a color index for a requested color, either the exact color or the closest possible alternative.

See also imagecolorclosestalpha().

> **Nota:** This function was added in PHP 4.0.6 and requires GD 2.0.1

imagecolorset (PHP 3, PHP 4 >= 4.0.0)

Set the color for the specified palette index

bool **imagecolorset** (resource image, int index, int red, int green, int blue) \linebreak

This sets the specified index in the palette to the specified color. This is useful for creating flood-fill-like effects in paletted images without the overhead of performing the actual flood-fill.

See also imagecolorat().

imagecolorsforindex (PHP 3, PHP 4 >= 4.0.0)

Get the colors for an index

array **imagecolorsforindex** (resource image, int index) \linebreak

This returns an associative array with red, green, and blue keys that contain the appropriate values for the specified color index.

See also imagecolorat() and imagecolorexact().

imagecolorstotal (PHP 3, PHP 4 >= 4.0.0)

Find out the number of colors in an image's palette

int **imagecolorstotal** (resource image) \linebreak

This returns the number of colors in the specified image's palette.

See also imagecolorat() and imagecolorsforindex().

imagecolortransparent (PHP 3, PHP 4 >= 4.0.0)

Define a color as transparent

int **imagecolortransparent** (resource image [, int color]) \linebreak

imagecolortransparent() sets the transparent color in the *image* image to *color*. *image* is the image identifier returned by imagecreate() and *color* is a color identifier returned by imagecolorallocate().

> **Nota:** The transparent color is a property of the image, transparency is not a property of the color. Once you have a set a color to be the transparent color, any regions of the image in that color that were drawn previously will be transparent.

The identifier of the new (or current, if none is specified) transparent color is returned.

imagecopy (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Copy part of an image

int **imagecopy** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h) \linebreak

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*.

imagecopymerge (PHP 4)

Copy and merge part of an image

int **imagecopymerge** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct) \linebreak

Copy a part of *src_im* onto *dst_im* starting at the x,y coordinates *src_x*, *src_y* with a width of *src_w* and a height of *src_h*. The portion defined will be copied onto the x,y coordinates, *dst_x* and *dst_y*. The two images will be merged according to *pct* which can range from 0 to 100. When *pct* = 0, no action is taken, when 100 this function behaves identically to imagecopy().

> **Nota:** This function was added in PHP 4.0.6

imagecopymergegray (PHP 4 >= 4.0.6)

Copy and merge part of an image with gray scale

int **imagecopymergegray** (resource dst_im, resource src_im, int dst_x, int dst_y, int src_x, int src_y, int src_w, int src_h, int pct) \linebreak

imagecopymergegray() copy a part of `src_im` onto `dst_im` starting at the x,y coordinates `src_x`, `src_y` with a width of `src_w` and a height of `src_h`. The portion defined will be copied onto the x,y coordinates, `dst_x` and `dst_y`. The two images will be merged according to `pct` which can range from 0 to 100. When `pct` = 0, no action is taken, when 100 this function behaves identically to imagecopy().

This function is identical to imagecopymerge() except that when merging it preservese the hue of the source by converting the destination pixels to gray scale before the copy operation.

Nota: This function was added in PHP 4.0.6

imagecopyresampled (PHP 4 >= 4.0.6)

Copy and resize part of an image with resampling

int **imagecopyresampled** (resource dst_im, resource src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH) \linebreak

imagecopyresampled() copies a rectangular portion of one image to another image, smoothly interpolating pixel values so that, in particular, reducing the size of an image still retains a great deal of clarity. `Dst_im` is the destination image, `src_im` is the source image identifier. If the source and destination coordinates and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if `dst_im` is the same as `src_im`) but if the regions overlap the results will be unpredictable.

See also imagecopyresized().

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagecopyresized (PHP 3, PHP 4 >= 4.0.0)

Copy and resize part of an image

int **imagecopyresized** (resource dst_im, resource src_im, int dstX, int dstY, int srcX, int srcY, int dstW, int dstH, int srcW, int srcH) \linebreak

imagecopyresized() copies a rectangular portion of one image to another image. `Dst_im` is the destination image, `src_im` is the source image identifier. If the source and destination coordinates

and width and heights differ, appropriate stretching or shrinking of the image fragment will be performed. The coordinates refer to the upper left corner. This function can be used to copy regions within the same image (if *dst_im* is the same as *src_im*) but if the regions overlap the results will be unpredictable.

See also imagecopyresampled().

imagecreate (PHP 3, PHP 4 >= 4.0.0)

Create a new palette based image

resource **imagecreate** (int x_size, int y_size) \linebreak

imagecreate() returns an image identifier representing a blank image of size *x_size* by *y_size*.

Esempio 1. Creating a new GD image stream and outputting an image.

```
<?php
header ("Content-type: image/png");
$im = @imagecreate (50, 100)
    or die ("Cannot Initialize new GD image stream");
$background_color = imagecolorallocate ($im, 255, 255, 255);
$text_color = imagecolorallocate ($im, 233, 14, 91);
imagestring ($im, 1, 5, 5,  "A Simple Text String", $text_color);
imagepng ($im);
?>
```

imagecreatefromgd (PHP 4 >= 4.1.0)

Create a new image from GD file or URL

resource **imagecreatefromgd** (string filename) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imagecreatefromgd2 (PHP 4 >= 4.1.0)

Create a new image from GD2 file or URL

resource **imagecreatefromgd2** (string filename) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imagecreatefromgd2part (PHP 4 >= 4.1.0)

Create a new image from a given part of GD2 file or URL

resource **imagecreatefromgd2part** (string filename, int srcX, int srcY, int width, int height) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imagecreatefromgif (PHP 3, PHP 4 >= 4.0.0)

Create a new image from file or URL

resource **imagecreatefromgif** (string filename) \linebreak

imagecreatefromgif() returns an image identifier representing the image obtained from the given filename.

imagecreatefromgif() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error GIF:

Esempio 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadGif ($imgname) {
    $im = @imagecreatefromgif ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im = imagecreate (150, 30); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

> **Nota:** Since all GIF support was removed from the GD library in version 1.6, this function is not available if you are using that version of the GD library.

imagecreatefromjpeg (PHP 3>= 3.0.16, PHP 4 >= 4.0.0)

Create a new image from file or URL

resource **imagecreatefromjpeg** (string filename) \linebreak

imagecreatefromjpeg() returns an image identifier representing the image obtained from the given filename.

imagecreatefromjpeg() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error JPEG:

Esempio 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadJpeg ($imgname) {
    $im = @imagecreatefromjpeg ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = imagecreate (150, 30); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

imagecreatefrompng (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Create a new image from file or URL

resource **imagecreatefrompng** (string filename) \linebreak

imagecreatefrompng() returns an image identifier representing the image obtained from the given filename.

imagecreatefrompng() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error PNG:

Esempio 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadPNG ($imgname) {
    $im = @imagecreatefrompng ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
        $im  = imagecreate (150, 30); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 150, 30, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

imagecreatefromstring (PHP 4 >= 4.0.4)

Create a new image from the image stream in the string

resource **imagecreatefromstring** (string image) \linebreak

imagecreatefromstring() returns an image identifier representing the image obtained from the given string.

imagecreatefromwbmp (PHP 4)

Create a new image from file or URL

resource **imagecreatefromwbmp** (string filename) \linebreak

imagecreatefromwbmp() returns an image identifier representing the image obtained from the given filename.

imagecreatefromwbmp() returns an empty string on failure. It also outputs an error message, which unfortunately displays as a broken link in a browser. To ease debugging the following example will produce an error WBMP:

Esempio 1. Example to handle an error during creation (courtesy vic@zymsys.com)

```
function LoadWBMP ($imgname) {
    $im = @imagecreatefromwbmp ($imgname); /* Attempt to open */
    if (!$im) { /* See if it failed */
```

```
        $im  = imagecreate (20, 20); /* Create a blank image */
        $bgc = imagecolorallocate ($im, 255, 255, 255);
        $tc  = imagecolorallocate ($im, 0, 0, 0);
        imagefilledrectangle ($im, 0, 0, 10, 10, $bgc);
        /* Output an errmsg */
        imagestring ($im, 1, 5, 5, "Error loading $imgname", $tc);
    }
    return $im;
}
```

Nota: WBMP support is only available if PHP was compiled against GD-1.8 or later.

imagecreatefromxbm (PHP 4)

Create a new image from file or URL

resource **imagecreatefromxbm** (string filename) \linebreak

imagecreatefromxbm() returns an image identifier representing the image obtained from the given filename.

imagecreatefromxpm (PHP 4)

Create a new image from file or URL

resource **imagecreatefromxpm** (string filename) \linebreak

imagecreatefromxpm() returns an image identifier representing the image obtained from the given filename.

imagecreatetruecolor (PHP 4 >= 4.0.6)

Create a new true color image

resource **imagecreatetruecolor** (int x_size, int y_size) \linebreak

imagecreatetruecolor() returns an image identifier representing a black image of size *x_size* by *y_size*.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagedashedline (PHP 3, PHP 4 >= 4.0.0)

Draw a dashed line

int **imagedashedline** (resource image, int x1, int y1, int x2, int y2, int col) \linebreak

This function is deprecated. Use combination of imagesetstyle() and imageline() instead.

imagedestroy (PHP 3, PHP 4 >= 4.0.0)

Destroy an image

int **imagedestroy** (resource image) \linebreak

imagedestroy() frees any memory associated with image `image`. `image` is the image identifier returned by the imagecreate() function.

imageellipse (PHP 4 >= 4.0.6)

Draw an ellipse

int **imageellipse** (resource image, int cx, int cy, int w, int h, int col) \linebreak

imageellipse() draws an ellipse centered at `cx`, `cy` (top left is 0, 0) in the image represented by `image`. `W` and `h` specifies the ellipse's width and height respectively. The color of the ellipse is specified by `color`.

> **Nota:** This function was added in PHP 4.0.6 and requires GD 2.0.2 or later

See also imagearc().

imagefill (PHP 3, PHP 4 >= 4.0.0)

Flood fill

int **imagefill** (resource image, int x, int y, int col) \linebreak

imagefill() performs a flood fill starting at coordinate `x`, `y` (top left is 0, 0) with color `col` in the image `image`.

imagefilledarc (PHP 4 >= 4.0.6)

Draw a partial ellipse and fill it

int **imagefilledarc** (resource image, int cx, int cy, int w, int h, int s, int e, int col, int style) \linebreak

imagefilledarc() draws a partial ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively while the start and end points are specified in degrees indicated by the *s* and *e* arguments. *style* is a bitwise OR of the following possibilities:

1. `IMG_ARC_PIE`
2. `IMG_ARC_CHORD`
3. `IMG_ARC_NOFILL`
4. `IMG_ARC_EDGED`

`IMG_ARC_PIE` and `IMG_ARC_CHORD` are mutually exclusive; `IMG_ARC_CHORD` just connects the starting and ending angles with a straight line, while `IMG_ARC_PIE` produces a rounded edge. `IMG_ARC_NOFILL` indicates that the arc or chord should be outlined, not filled. `IMG_ARC_EDGED`, used together with `IMG_ARC_NOFILL`, indicates that the beginning and ending angles should be connected to the center - this is a good way to outline (rather than fill) a 'pie slice'.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1

imagefilledellipse (PHP 4 >= 4.0.6)

Draw a filled ellipse

int **imagefilledellipse** (resource image, int cx, int cy, int w, int h, int col) \linebreak

imagefilledellipse() draws an ellipse centered at *cx*, *cy* (top left is 0, 0) in the image represented by *image*. *W* and *h* specifies the ellipse's width and height respectively. The ellipse is filled using *color*

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

See also imagefilledarc().

imagefilledpolygon (PHP 3, PHP 4 >= 4.0.0)

Draw a filled polygon

int **imagefilledpolygon** (resource image, array points, int num_points, int col) \linebreak

imagefilledpolygon() creates a filled polygon in image *image*. *points* is a PHP array containing the polygon's vertices, ie. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. *num_points* is the total number of vertices.

imagefilledrectangle (PHP 3, PHP 4 >= 4.0.0)

Draw a filled rectangle

int **imagefilledrectangle** (resource image, int x1, int y1, int x2, int y2, int col) \linebreak

imagefilledrectangle() creates a filled rectangle of color *col* in image *image* starting at upper left coordinates *x1*, *y1* and ending at bottom right coordinates *x2*, *y2*. 0, 0 is the top left corner of the image.

imagefilltoborder (PHP 3, PHP 4 >= 4.0.0)

Flood fill to specific color

int **imagefilltoborder** (resource image, int x, int y, int border, int col) \linebreak

imagefilltoborder() performs a flood fill whose border color is defined by *border*. The starting point for the fill is *x*, *y* (top left is 0, 0) and the region is filled with color *col*.

imagefontheight (PHP 3, PHP 4 >= 4.0.0)

Get font height

int **imagefontheight** (int font) \linebreak

Returns the pixel height of a character in the specified font.

See also imagefontwidth() and imageloadfont().

imagefontwidth (PHP 3, PHP 4 >= 4.0.0)

Get font width

int **imagefontwidth** (int font) \linebreak

Returns the pixel width of a character in font.

See also imagefontheight() and imageloadfont().

imageftbbox (PHP 4 >= 4.1.0)

Give the bounding box of a text using fonts via freetype2

array **imageftbbox** (int size, int angle, string font_file, string text [, array extrainfo]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imagefttext (PHP 4 >= 4.1.0)

Write text to the image using fonts using FreeType 2

array **imagefttext** (resource image, int size, int angle, int x, int y, int col, string font_file, string text [, array extrainfo]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imagegammacorrect (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Apply a gamma correction to a GD image

int **imagegammacorrect** (resource image, float inputgamma, float outputgamma) \linebreak

The **imagegammacorrect()** function applies gamma correction to a gd image stream (*image*) given an input gamma, the parameter *inputgamma* and an output gamma, the parameter *outputgamma*.

imagegd (PHP 4 >= 4.1.0)

Output GD image to browser or file

int **imagegd** (resource image [, string filename]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imagegd2 (PHP 4 >= 4.1.0)

Output GD2 image to browser or file

int **imagegd2** (resource image [, string filename]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imagegif (PHP 3, PHP 4 >= 4.0.0)

Output image to browser or file

int **imagegif** (resource image [, string filename]) \linebreak

imagegif() creates the GIF file in filename from the image *image*. The *image* argument is the return from the imagecreate() function.

The image format will be GIF87a unless the image has been made transparent with imagecolortransparent(), in which case the image format will be GIF89a.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/gif content-type using header(), you can create a PHP script that outputs GIF images directly.

> **Nota:** Since all GIF support was removed from the GD library in version 1.6, this function is not available if you are using that version of the GD library.
>
> The following code snippet allows you to write more portable PHP applications by auto-detecting the type of GD support which is available. Replace the sequence `header ("Content-type: image/gif"); imagegif ($im);` by the more flexible sequence:

```
<?php
if (function_exists("imagegif")) {
    header ("Content-type: image/gif");
    imagegif ($im);
}
elseif (function_exists("imagejpeg")) {
    header ("Content-type: image/jpeg");
    imagejpeg ($im, "", 0.5);
}
elseif (function_exists("imagepng")) {
    header ("Content-type: image/png");
    imagepng ($im);
}
elseif (function_exists("imagewbmp")) {
    header ("Content-type: image/vnd.wap.wbmp");
    imagewbmp ($im);
}
else
    die("No image support in this PHP server");
```

```
?>
```

Nota: As of version 3.0.18 and 4.0.2 you can use the function imagetypes() in place of function_exists() for checking the presence of the various supported image formats:

```
if (imagetypes() & IMG_GIF) {
    header ("Content-type: image/gif");
    imagegif ($im);
}
elseif (imagetypes() & IMG_JPG) {
        ... etc.
```

See also imagepng(), imagewbmp(), imagejpeg(), imagetypes().

imageinterlace (PHP 3, PHP 4 >= 4.0.0)

Enable or disable interlace

int **imageinterlace** (resource image [, int interlace]) \linebreak

imageinterlace() turns the interlace bit on or off. If interlace is 1 the image will be interlaced, and if interlace is 0 the interlace bit is turned off.

If the interlace bit is set and the image is used as a JPEG image, the image is created as a progressive JPEG.

This function returns whether the interlace bit is set for the image.

imagejpeg (PHP 3>= 3.0.16, PHP 4 >= 4.0.0)

Output image to browser or file

int **imagejpeg** (resource image [, string filename [, int quality]]) \linebreak

imagejpeg() creates the JPEG file in filename from the image `image`. The `image` argument is the return from the imagecreate() function.

The filename argument is optional, and if left off, the raw image stream will be output directly. To skip the filename argument in order to provide a quality argument just use an empty string (”). By

sending an image/jpeg content-type using header(), you can create a PHP script that outputs JPEG images directly.

> **Nota:** JPEG support is only available if PHP was compiled against GD-1.8 or later.

`quality` is optional, and ranges from 0 (worst quality, smaller file) to 100 (best quality, biggest file). The default is the default IJG quality value (about 75).

If you want to output Progressive JPEGs, you need to set interlacing on with imageinterlace().

See also imagepng(), imagegif(), imagewbmp(), imageinterlace() and imagetypes().

imageline (PHP 3, PHP 4 >= 4.0.0)

Draw a line

int **imageline** (resource image, int x1, int y1, int x2, int y2, int col) \linebreak

imageline() draws a line from `x1`, `y1` to `x2`, `y2` (top left is 0, 0) in image im of color `col`.

See also imagecreate() and imagecolorallocate().

imageloadfont (PHP 3, PHP 4 >= 4.0.0)

Load a new font

int **imageloadfont** (string file) \linebreak

imageloadfont() loads a user-defined bitmap font and returns an identifier for the font (that is always greater than 5, so it will not conflict with the built-in fonts).

The font file format is currently binary and architecture dependent. This means you should generate the font files on the same type of CPU as the machine you are running PHP on.

Tabella 1. Font file format

byte position	C data type	description
byte 0-3	int	number of characters in the font
byte 4-7	int	value of first character in the font (often 32 for space)
byte 8-11	int	pixel width of each character
byte 12-15	int	pixel height of each character
byte 16-	char	array with character data, one byte per pixel in each character, for a total of (nchars*width*height) bytes.

See also imagefontwidth() and imagefontheight().

imagepalettecopy (PHP 4)

Copy the palette from one image to another

int **imagepalettecopy** (resource destination, resource source) \linebreak

imagepalettecopy() copies the palette from the *source* image to the *destination* image.

imagepng (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Output a PNG image to either the browser or a file

int **imagepng** (resource image [, string filename]) \linebreak

The **imagepng()** outputs a GD image stream (*image*) in PNG format to standard output (usually the browser) or, if a filename is given by the *filename* it outputs the image to the file.

```
<?php
$im = imagecreatefrompng ("test.png");
imagepng ($im);
?>
```

See also imagegif(), imagewbmp(), imagejpeg(), imagetypes().

imagepolygon (PHP 3, PHP 4 >= 4.0.0)

Draw a polygon

int **imagepolygon** (resource image, array points, int num_points, int col) \linebreak

imagepolygon() creates a polygon in image id. *points* is a PHP array containing the polygon's vertices, ie. points[0] = x0, points[1] = y0, points[2] = x1, points[3] = y1, etc. *num_points* is the total number of vertices.

See also imagecreate().

imagepsbbox (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Give the bounding box of a text rectangle using PostScript Type1 fonts

array **imagepsbbox** (string text, int font, int size [, int space [, int tightness [, float angle]]]) \linebreak

Size is expressed in pixels.

Space allows you to change the default value of a space in a font. This amount is added to the normal value and can also be negative.

Tightness allows you to control the amount of white space between characters. This amount is added to the normal character width and can also be negative.

Angle is in degrees.

Parameters *space* and *tightness* are expressed in character space units, where 1 unit is 1/1000th of an em-square.

Parameters *space*, *tightness*, and *angle* are optional.

The bounding box is calculated using information available from character metrics, and unfortunately tends to differ slightly from the results achieved by actually rasterizing the text. If the angle is 0 degrees, you can expect the text to need 1 pixel more to every direction.

This function returns an array containing the following elements:

0	lower left x-coordinate
1	lower left y-coordinate
2	upper right x-coordinate
3	upper right y-coordinate

See also imagepstext().

imagepscopyfont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Make a copy of an already loaded font for further modification

int **imagepscopyfont** (int fontindex) \linebreak

Use this function if you need make further modifications to the font, for example extending/condensing, slanting it or changing it's character encoding vector, but need to keep the original along as well. Note that the font you want to copy must be one obtained using imagepsloadfont(), not a font that is itself a copied one. You can although make modifications to it before copying.

If you use this function, you *must* free the fonts obtained this way yourself and in reverse order. Otherwise your script *will* hang.

In the case everything went right, a valid font index will be returned and can be used for further purposes. Otherwise the function returns FALSE and prints a message describing what went wrong.

See also imagepsloadfont().

imagepsencodefont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Change the character encoding vector of a font

int **imagepsencodefont** (int font_index, string encodingfile) \linebreak

Loads a character encoding vector from from a file and changes the fonts encoding vector to it. As a PostScript fonts default vector lacks most of the character positions above 127, you'll definitely want to change this if you use an other language than english. The exact format of this file is described in T1libs documentation. T1lib comes with two ready-to-use files, IsoLatin1.enc and IsoLatin2.enc.

If you find yourself using this function all the time, a much better way to define the encoding is to set ps.default_encoding in the configuration file to point to the right encoding file and all fonts you load will automatically have the right encoding.

imagepsextendfont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Extend or condense a font

bool **imagepsextendfont** (int font_index, float extend) \linebreak

Extend or condense a font (*font_index*), if the value of the *extend* parameter is less than one you will be condensing the font.

imagepsfreefont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Free memory used by a PostScript Type 1 font

void **imagepsfreefont** (int fontindex) \linebreak

See also imagepsloadfont().

imagepsloadfont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Load a PostScript Type 1 font from file

int **imagepsloadfont** (string filename) \linebreak

In the case everything went right, a valid font index will be returned and can be used for further purposes. Otherwise the function returns FALSE and prints a message describing what went wrong, which you cannot read directly, while the output type is image.

```
<?php
header ("Content-type: image/jpeg");
$im = imagecreate (350, 45);
$black = imagecolorallocate ($im, 0, 0, 0);
$white = imagecolorallocate ($im, 255, 255, 255);
$font = imagepsloadfont ("bchbi.pfb"); // or locate your .pfb files on your machine
```

```
imagepstext ($im, "Testing... It worked!", $font, 32, $white, $black, 32, 32);
imagepsfreefont ($font);
imagejpeg ($im, "", 100); //for best quality...your mileage may vary
imagedestroy ($im);
?>
```

See also imagepsfreefont().

imagepsslantfont (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Slant a font

bool **imagepsslantfont** (int font_index, float slant) \linebreak

Slant a font given by the *font_index* parameter with a slant of the value of the *slant* parameter.

imagepstext (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

To draw a text string over an image using PostScript Type1 fonts

array **imagepstext** (resource image, string text, int font, int size, int foreground, int background, int x, int y [, int space [, int tightness [, float angle [, int antialias_steps]]]]) \linebreak

Size is expressed in pixels.

Foreground is the color in which the text will be painted. *Background* is the color to which the text will try to fade in with antialiasing. No pixels with the color *background* are actually painted, so the background image does not need to be of solid color.

The coordinates given by *x*, *y* will define the origin (or reference point) of the first character (roughly the lower-left corner of the character). This is different from the imagestring(), where *x*, *y* define the upper-right corner of the first character. Refer to PostScipt documentation about fonts and their measuring system if you have trouble understanding how this works.

Space allows you to change the default value of a space in a font. This amount is added to the normal value and can also be negative.

Tightness allows you to control the amount of white space between characters. This amount is added to the normal character width and can also be negative.

Angle is in degrees.

Antialias_steps allows you to control the number of colours used for antialiasing text. Allowed values are 4 and 16. The higher value is recommended for text sizes lower than 20, where the effect in text quality is quite visible. With bigger sizes, use 4. It's less computationally intensive.

Parameters *space* and *tightness* are expressed in character space units, where 1 unit is 1/1000th of an em-square.

Parameters *space*, *tightness*, *angle* and *antialias* are optional.

This function returns an array containing the following elements:

0	lower left x-coordinate
1	lower left y-coordinate
2	upper right x-coordinate
3	upper right y-coordinate

See also imagepsbbox().

imagerectangle (PHP 3, PHP 4 >= 4.0.0)

Draw a rectangle

int **imagerectangle** (resource image, int x1, int y1, int x2, int y2, int col) \linebreak

imagerectangle() creates a rectangle of color col in image *image* starting at upper left coordinate x1, y1 and ending at bottom right coordinate x2, y2. 0, 0 is the top left corner of the image.

imagesetbrush (PHP 4 >= 4.0.6)

Set the brush image for line drawing

int **imagesetbrush** (resource image, resource brush) \linebreak

imagesetbrush() sets the brush image to be used by all line drawing functions (such as imageline() and imagepolygon()) when drawing with the special colors `IMG_COLOR_BRUSHED` or `IMG_COLOR_STYLEDBRUSHED`.

> **Nota:** You need not take special action when you are finished with a brush, but if you destroy the brush image, you must not use the `IMG_COLOR_BRUSHED` or `IMG_COLOR_STYLEDBRUSHED` colors until you have set a new brush image!

> **Nota:** This function was added in PHP 4.0.6

imagesetpixel (PHP 3, PHP 4 >= 4.0.0)

Set a single pixel

int **imagesetpixel** (resource image, int x, int y, int col) \linebreak

imagesetpixel() draws a pixel at *x*, *y* (top left is 0, 0) in image *image* of color *col*.

See also imagecreate() and imagecolorallocate().

imagesetstyle (PHP 4 >= 4.0.6)

Set the style for line drawing

int **imagesetstyle** (resource image, array style) \linebreak

imagesetstyle() sets the style to be used by all line drawing functions (such as imageline() and imagepolygon()) when drawing with the special color `IMG_COLOR_STYLED` or lines of images with color `IMG_COLOR_STYLEDBRUSHED`.

The *`style`* parameter is an array of pixels. Following example script draws a dashed line from upper left to lower right corner of the canvas:

Esempio 1. imagesetstyle

```
<?php
header ("Content-type: image/png");
$im  = imagecreate (100, 100);
$w   = imagecolorallocate ($im, 255, 255, 255);
$red = imagecolorallocate ($im, 255, 0, 0);

/* Draw a dashed line, 5 red pixels, 5 white pixels */
$style = array ($red,$red,$red,$red,$red,$w,$w,$w,$w,$w);
imagesetstyle ($im, $style);
imageline ($im, 0, 0, 100, 100, IMG_COLOR_STYLED);

/* Draw a line of happy faces using imagesetbrush() with imagesetstyle */
$style = array ($w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$w,$red);
imagesetstyle ($im, $style);

$brush = imagecreatefrompng ("http://www.libpng.org/pub/png/images/smile.happy.png");
imagecolortransparent ($brush, $w);
imagesetbrush ($im, $brush);
imageline ($im, 100, 0, 0, 100, IMG_COLOR_STYLEDBRUSHED);

imagepng ($im);
imagedestroy ($im);
?>
```

See also imagesetbrush(), imageline().

> **Nota:** This function was added in PHP 4.0.6

imagesetthickness (PHP 4 >= 4.0.6)

Set the thickness for line drawing

void **imagesetthickness** (resource image, int thickness) \linebreak

imagesetthickness() sets the thickness of the lines drawn when drawing rectangles, polygons, ellipses etc. etc. to *thickness* pixels.

Nota: This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagesettile (PHP 4 >= 4.0.6)

Set the tile image for filling

int **imagesettile** (resource image, resource tile) \linebreak

imagesettile() sets the tile image to be used by all region filling functions (such as imagefill() and imagefilledpolygon()) when filling with the special color `IMG_COLOR_TILED`.

A tile is an image used to fill an area with a repeated pattern. *Any* GD image can be used as a tile, and by setting the transparent color index of the tile image with imagecolortransparent(), a tile allows certain parts of the underlying area to shine through can be created.

Nota: You need not take special action when you are finished with a tile, but if you destroy the tile image, you must not use the `IMG_COLOR_TILED` color until you have set a new tile image!

Nota: This function was added in PHP 4.0.6

imagestring (PHP 3, PHP 4 >= 4.0.0)

Draw a string horizontally

int **imagestring** (resource image, int font, int x, int y, string s, int col) \linebreak

imagestring() draws the string *s* in the image identified by *image* at coordinates *x*, *y* (top left is 0, 0) in color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also imageloadfont().

imagestringup (PHP 3, PHP 4 >= 4.0.0)

Draw a string vertically

int **imagestringup** (resource image, int font, int x, int y, string s, int col) \linebreak

imagestringup() draws the string *s* vertically in the image identified by *image* at coordinates *x*, *y* (top left is 0, 0) in color *col*. If font is 1, 2, 3, 4 or 5, a built-in font is used.

See also imageloadfont().

imagesx (PHP 3, PHP 4 >= 4.0.0)

Get image width

int **imagesx** (resource image) \linebreak

imagesx() returns the width of the image identified by *image*.

See also imagecreate() and imagesy().

imagesy (PHP 3, PHP 4 >= 4.0.0)

Get image height

int **imagesy** (resource image) \linebreak

imagesy() returns the height of the image identified by *image*.

See also imagecreate() and imagesx().

imagetruecolortopalette (PHP 4 >= 4.0.6)

Convert a true color image to a palette image

void **imagetruecolortopalette** (resource image, bool dither, int ncolors) \linebreak

imagetruecolortopalette() converts a truecolor image to a palette image. The code for this function was originally drawn from the Independent JPEG Group library code, which is excellent. The code has been modified to preserve as much alpha channel information as possible in the resulting palette, in addition to preserving colors as well as possible. This does not work as well as might be hoped. It is usually best to simply produce a truecolor output image instead, which guarantees the highest output quality.

dither indicates if the image should be dithered - if it is TRUE then dithering will be used which will result in a more speckled image but with better color approximation.

ncolors sets the maximum number of colors that should be retained in the palette.

> **Nota:** This function was added in PHP 4.0.6 and requires GD 2.0.1 or later

imagettfbbox (PHP 3>= 3.0.1, PHP 4 >= 4.0.0)

Give the bounding box of a text using TrueType fonts

array **imagettfbbox** (int size, int angle, string fontfile, string text) \linebreak

This function calculates and returns the bounding box in pixels for a TrueType text.

text

The string to be measured.

size

The font size in pixels.

fontfile

The name of the TrueType font file. (Can also be an URL.) Depending on which version of the GD library that PHP is using, it may attempt to search for files that do not begin with a leading '/' by appending '.ttf' to the filename and searching along a library-defined font path.

angle

Angle in degrees in which *text* will be measured.

imagettfbbox() returns an array with 8 elements representing four points making the bounding box of the text:

0	lower left corner, X position
1	lower left corner, Y position
2	lower right corner, X position
3	lower right corner, Y position
4	upper right corner, X position
5	upper right corner, Y position
6	upper left corner, X position
7	upper left corner, Y position

The points are relative to the *text* regardless of the angle, so "upper left" means in the top left-hand corner seeing the text horizontallty.

This function requires both the GD library and the FreeType library.

See also imagettftext().

imagettftext (PHP 3, PHP 4 >= 4.0.0)

Write text to the image using TrueType fonts

array **imagettftext** (resource image, int size, int angle, int x, int y, int col, string fontfile, string text) \linebreak

imagettftext() draws the string *text* in the image identified by *image*, starting at coordinates *x*, *y* (top left is 0, 0), at an angle of *angle* in color *col*, using the TrueType font file identified by *fontfile*. Depending on which version of the GD library that PHP is using, when *fontfile* does not begin with a leading '/', '.ttf' will be appended to the filename and the the library will attempt to search for that filename along a library-defined font path.

The coordinates given by `x`, `y` will define the basepoint of the first character (roughly the lower-left corner of the character). This is different from the imagestring(), where x, y define the upper-right corner of the first character.

`Angle` is in degrees, with 0 degrees being left-to-right reading text (3 o'clock direction), and higher values representing a counter-clockwise rotation. (i.e., a value of 90 would result in bottom-to-top reading text).

`Fontfile` is the path to the TrueType font you wish to use.

`Text` is the text string which may include UTF-8 character sequences (of the form: {) to access characters in a font beyond the first 255.

`Col` is the color index. Using the negative of a color index has the effect of turning off antialiasing.

imagettftext() returns an array with 8 elements representing four points making the bounding box of the text. The order of the points is lower left, lower right, upper right, upper left. The points are relative to the text regardless of the angle, so "upper left" means in the top left-hand corner when you see the text horizontallty.

This example script will produce a black GIF 400x30 pixels, with the words "Testing..." in white in the font Arial.

Esempio 1. imagettftext

```
<?php
header ("Content-type: image/gif");
$im = imagecreate (400, 30);
$black = imagecolorallocate ($im, 0, 0, 0);
$white = imagecolorallocate ($im, 255, 255, 255);
imagettftext ($im, 20, 0, 10, 20, $white, "/path/arial.ttf", "Testing...Omega: &#937;");
imagegif ($im);
imagedestroy ($im);
?>
```

This function requires both the GD library and the FreeType (http://www.freetype.org/) library.

See also imagettfbbox().

imagetypes (PHP 3 CVS only, PHP 4 >= 4.0.2)

Return the image types supported by this PHP build

int **imagetypes** (void) \linebreak

This function returns a bit-field corresponding to the image formats supported by the version of GD linked into PHP. The following bits are returned, `IMG_GIF` | `IMG_JPG` | `IMG_PNG` | `IMG_WBMP`. To check for PNG support, for example, do this:

Esempio 1. imagetypes

```
<?php
if (imagetypes() & IMG_PNG) {
    echo "PNG Support is enabled";
}
?>
```

imagewbmp (PHP 3>= 3.0.15, PHP 4)

Output image to browser or file

int **imagewbmp** (resource image [, string filename [, int foreground]]) \linebreak

imagewbmp() creates the WBMP file in filename from the image *image*. The *image* argument is the return from the imagecreate() function.

The filename argument is optional, and if left off, the raw image stream will be output directly. By sending an image/vnd.wap.wbmp content-type using header(), you can create a PHP script that outputs WBMP images directly.

> **Nota:** WBMP support is only available if PHP was compiled against GD-1.8 or later.

Using the optional *foreground* parameter, you can set the foreground color. Use an identifier obtained from imagecolorallocate(). The default foreground color is black.

See also image2wbmp(), imagepng(), imagegif(), imagejpeg(), imagetypes().

iptcembed (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Embed binary IPTC data into a JPEG image

array **iptcembed** (string iptcdata, string jpeg_file_name [, int spool]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

jpeg2wbmp (PHP 4 >= 4.0.5)

Convert JPEG image file to WBMP image file

int **jpeg2wbmp** (string jpegname, string wbmpname, int d_height, int d_width, int threshold) \linebreak

Converts the *jpegname* JPEG file to WBMP format, and saves it as *wbmpname*. With the *d_height* and *d_width* you specify the height and width of the destination image.

> **Nota:** WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also png2wbmp().

png2wbmp (PHP 4 >= 4.0.5)

Convert PNG image file to WBMP image file

int **png2wbmp** (string pngname, string wbmpname, int d_height, int d_width, int threshold) \linebreak

Converts the *pngname* PNG file to WBMP format, and saves it as *wbmpname*. With the *d_height* and *d_width* you specify the height and width of the destination image.

> **Nota:** WBMP support is only available if PHP was compiled against GD-1.8 or later.

See also jpeg2wbmp().

read_exif_data (PHP 4)

Reads header information stored in TIFF and JPEG images

array **exif_read_data** (string filename, string sections, bool arrays, bool thumbnail) \linebreak

> **Nota:** The **read_exif_data()** function is an alias for exif_read_data().

See also exif_thumbnail().

XLIII. IMAP, POP3 and NNTP functions

To get these functions to work, you have to compile PHP with `--with-imap`. That requires the c-client library to be installed. Grab the latest version from ftp://ftp.cac.washington.edu/imap/ and compile it.

Then copy `c-client/c-client.a` to `/usr/local/lib/libc-client.a` or some other directory on your link path and copy `c-client/*.h` to `/usr/local/include` or some other directory in your include path.

> **Nota:** Depending how the c-client was configured, you might also need to add `--with-imap-ssl=/path/to/openssl/` and/or `--with-kerberos` into the PHP configure line.

Note that these functions are not limited to the IMAP protocol, despite their name. The underlying c-client library also supports NNTP, POP3 and local mailbox access methods.

This document can't go into detail on all the topics touched by the provided functions. Further information is provided by the documentation of the c-client library source (`docs/internal.txt`). and the following RFC documents:

- RFC2821 (http://www.faqs.org/rfcs/rfc2821.html): Simple Mail Transfer Protocol (SMTP).
- RFC2822 (http://www.faqs.org/rfcs/rfc2822.html): Standard for ARPA internet text messages.
- RFC2060 (http://www.faqs.org/rfcs/rfc2060.html): Internet Message Access Protocol (IMAP) Version 4rev1.
- RFC1939 (http://www.faqs.org/rfcs/rfc1939.html): Post Office Protocol Version 3 (POP3).
- RFC977 (http://www.faqs.org/rfcs/rfc977.html): Network News Transfer Protocol (NNTP).
- RFC2076 (http://www.faqs.org/rfcs/rfc2076.html): Common Internet Message Headers.
- RFC2045 (http://www.faqs.org/rfcs/rfc2045.html) , RFC2046 (http://www.faqs.org/rfcs/rfc2046.html) , RFC2047 (http://www.faqs.org/rfcs/rfc2047.html) , RFC2048 (http://www.faqs.org/rfcs/rfc2048.html) & RFC2049 (http://www.faqs.org/rfcs/rfc2049.html): Multipurpose Internet Mail Extensions (MIME).

A detailed overview is also available in the books Programming Internet Email (http://www.oreilly.com/catalog/progintemail/noframes.html) by David Wood and Managing IMAP (http://www.oreilly.com/catalog/mimap/noframes.html) by Dianna Mullet & Kevin Mullet.

imap_8bit (PHP 3, PHP 4 >= 4.0.0)

Convert an 8bit string to a quoted-printable string

string **imap_8bit** (string string) \linebreak

Convert an 8bit string to a quoted-printable string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), section 6.7).

Returns a quoted-printable string.

See also imap_qprint().

imap_alerts (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

This function returns all IMAP alert messages (if any) that have occurred during this page request or since the alert stack was reset

array **imap_alerts** (void) \linebreak

This function returns an array of all of the IMAP alert messages generated since the last **imap_alerts()** call, or the beginning of the page. When **imap_alerts()** is called, the alert stack is subsequently cleared. The IMAP specification requires that these messages be passed to the user.

imap_append (PHP 3, PHP 4 >= 4.0.0)

Append a string message to a specified mailbox

int **imap_append** (int imap_stream, string mbox, string message [, string flags]) \linebreak

Returns `TRUE` on sucess, `FALSE` on error.

imap_append() appends a string message to the specified mailbox *mbox*. If the optional *flags* is specified, writes the *flags* to that mailbox also.

When talking to the Cyrus IMAP server, you must use "\r\n" as your end-of-line terminator instead of "\n" or the operation will fail.

Esempio 1. imap_append() example

```
$stream = imap_open("{your.imap.host}INBOX.Drafts","username", "password");

$check = imap_check($stream);
print "Msg Count before append: ". $check->Nmsgs."\n";

imap_append($stream,"{your.imap.host}INBOX.Drafts"
                   ,"From: me@my.host\r\n"
                   ."To: you@your.host\r\n"
                   ."Subject: test\r\n"
                   ."\r\n"
                   ."this is a test message, please ignore\r\n"
```

```
                  );

$check = imap_check($stream);
print "Msg Count after append : ". $check->Nmsgs."\n";

imap_close($stream);
```

imap_base64 (PHP 3, PHP 4 >= 4.0.0)

Decode BASE64 encoded text

string **imap_base64** (string text) \linebreak

imap_base64() function decodes BASE-64 encoded text (see RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Section 6.8). The decoded message is returned as a string.

See also imap_binary().

imap_binary (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Convert an 8bit string to a base64 string

string **imap_binary** (string string) \linebreak

Convert an 8bit string to a base64 string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), Section 6.8).

Returns a base64 string.

See also imap_base64().

imap_body (PHP 3, PHP 4 >= 4.0.0)

Read the message body

string **imap_body** (int imap_stream, int msg_number [, int flags]) \linebreak

imap_body() returns the body of the message, numbered *msg_number* in the current mailbox. The optional *flags* are a bit mask with one or more of the following:

- FT_UID - The *msgno* is a UID
- FT_PEEK - Do not set the \Seen flag if not already set
- FT_INTERNAL - The return string is in internal format, will not canonicalize to CRLF.

imap_body() will only return a verbatim copy of the message body. To extract single parts of a multipart MIME-encoded message you have to use imap_fetchstructure() to analyze its structure and imap_fetchbody() to extract a copy of a single body component.

imap_bodystruct (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Read the structure of a specified body section of a specific message

object **imap_bodystruct** (int stream_id, int msg_no, int section) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imap_check (PHP 3, PHP 4 >= 4.0.0)

Check current mailbox

object **imap_check** (int imap_stream) \linebreak

Returns information about the current mailbox. Returns FALSE on failure.

The **imap_check()** function checks the current mailbox status on the server and returns the information in an object with following properties:

- Date - last change of mailbox contents
- Driver - protocol used to access this mailbox: POP3, IMAP, NNTP
- Mailbox - the mailbox name
- Nmsgs - number of messages in the mailbox
- Recent - number of recent messages in the mailbox

imap_clearflag_full (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Clears flags on messages

string **imap_clearflag_full** (int stream, string sequence, string flag, string options) \linebreak

This function causes a store to delete the specified flag to the flags set for the messages in the specified sequence. The flags which you can unset are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", and "\\Recent" (as defined by RFC2060).

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

imap_close (PHP 3, PHP 4 >= 4.0.0)

Close an IMAP stream

int **imap_close** (int imap_stream [, int flags]) \linebreak

Close the imap stream. Takes an optional *flag* CL_EXPUNGE, which will silently expunge the mailbox before closing, removing all messages marked for deletion.

imap_createmailbox (PHP 3, PHP 4 >= 4.0.0)

Create a new mailbox

int **imap_createmailbox** (int imap_stream, string mbox) \linebreak

imap_createmailbox() creates a new mailbox specified by *mbox*. Names containing international characters should be encoded by imap_utf7_encode()

Returns TRUE on success and FALSE on error.

See also imap_renamemailbox(), imap_deletemailbox() and imap_open() for the format of *mbox* names.

Esempio 1. imap_createmailbox() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
     or die("can’t connect: ".imap_last_error());

$name1 = "phpnewbox";
$name2 = imap_utf7_encode("phpnewb&ouml;x");

$newname = $name1;

echo "Newname will be ’$name1’<br>\n";

# we will now create a new mailbox "phptestbox" in your inbox folder,
# check its status after creation and finaly remove it to restore
# your inbox to its initial state
if(@imap_createmailbox($mbox,imap_utf7_encode("{your.imap.host}INBOX.$newname"))) {
  $status = @imap_status($mbox,"{your.imap.host}INBOX.$newname",SA_ALL);
  if($status) {
    print("your new mailbox ’$name1’ has the following status:<br>\n");
    print("Messages:   ". $status->messages   )."<br>\n";
```

```
    print("Recent:     ". $status->recent     )."<br>\n";
    print("Unseen:     ". $status->unseen     )."<br>\n";
    print("UIDnext:    ". $status->uidnext    )."<br>\n";
    print("UIDvalidity:". $status->uidvalidity)."<br>\n";

    if(imap_renamemailbox($mbox,"{your.imap.host}INBOX.$newname","{your.imap.host}INBOX.$
      echo "renamed new mailbox from '$name1' to '$name2'<br>\n";
      $newname=$name2;
    } else {
      print "imap_renamemailbox on new mailbox failed: ".imap_last_error()."<br>\n";
    }
  } else {
    print  "imap_status on new mailbox failed: ".imap_last_error()."<br>\n";
  }
  if(@imap_deletemailbox($mbox,"{your.imap.host}INBOX.$newname")) {
    print "new mailbox removed to restore initial state<br>\n";
  } else {
    print  "imap_deletemailbox on new mailbox failed: ".implode("<br>\n",imap_errors())."
  }

} else {
  print  "could not create new mailbox: ".implode("<br>\n",imap_errors())."<br>\n";
}

imap_close($mbox);
```

imap_delete (PHP 3, PHP 4 >= 4.0.0)

Mark a messge for deletion from current mailbox

int **imap_delete** (int imap_stream, int msg_number [, int flags]) \linebreak

Returns `TRUE`.

imap_delete() marks messages listed in *`msg_number`* for deletion. The optional *`flags`* parameter only has a single option, *`FT_UID`*, which tells the function to treat the *`msg_number`* argument as a *`UID`*. Messages marked for deletion will stay in the mailbox until either imap_expunge() is called or imap_close() is called with the optional parameter CL_EXPUNGE.

> **Nota:** POP3 mailboxes do not have their message flags saved between connections, so imap_expunge() must be called during the same connection in order for messages marked for deletion to actually be purged.

Esempio 1. imap_delete() Beispiel

```
$mbox = imap_open ("{your.imap.host}INBOX", "username", "password")
    or die ("can’t connect: " . imap_last_error());

$check = imap_mailboxmsginfo ($mbox);
print "Messages before delete: " . $check->Nmsgs . "<br>\n" ;
imap_delete ($mbox, 1);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after  delete: " . $check->Nmsgs . "<br>\n" ;
imap_expunge ($mbox);
$check = imap_mailboxmsginfo ($mbox);
print "Messages after expunge: " . $check->Nmsgs . "<br>\n" ;
imap_close ($mbox);
```

imap_deletemailbox (PHP 3, PHP 4 >= 4.0.0)

Delete a mailbox

int **imap_deletemailbox** (int imap_stream, string mbox) \linebreak

imap_deletemailbox() deletes the specified mailbox (see imap_open() for the format of *mbox* names).

Returns TRUE on success and FALSE on error.

See also imap_createmailbox(), imap_renamemailbox(), and imap_open() for the format of *mbox*.

imap_errors (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

This function returns all of the IMAP errors (if any) that have occurred during this page request or since the error stack was reset.

array **imap_errors** (void) \linebreak

This function returns an array of all of the IMAP error messages generated since the last **imap_errors()** call, or the beginning of the page. When **imap_errors()** is called, the error stack is subsequently cleared.

imap_expunge (PHP 3, PHP 4 >= 4.0.0)

Delete all messages marked for deletion

int **imap_expunge** (int imap_stream) \linebreak

imap_expunge() deletes all the messages marked for deletion by imap_delete(), imap_mail_move(), or imap_setflag_full().

Returns `TRUE`.

imap_fetch_overview (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Read an overview of the information in the headers of the given message

array **imap_fetch_overview** (int imap_stream, string sequence [, int flags]) \linebreak

This function fetches mail headers for the given *`sequence`* and returns an overview of their contents. *`sequence`* will contain a sequence of message indices or UIDs, if *`flags`* contains FT_UID. The returned value is an array of objects describing one message header each:

- subject - the messages subject
- from - who sent it
- date - when was it sent
- message_id - Message-ID
- references - is a reference to this message id
- size - size in bytes
- uid - UID the message has in the mailbox
- msgno - message sequence number in the maibox
- recent - this message is flagged as recent
- flagged - this message is flagged
- answered - this message is flagged as answered
- deleted - this message is flagged for deletion
- seen - this message is flagged as already read
- draft - this message is flagged as being a draft

Esempio 1. imap_fetch_overview() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     or die("can't connect: ".imap_last_error());

$overview = imap_fetch_overview($mbox,"2,4:6",0);

if(is_array($overview)) {
        reset($overview);
        while( list($key,$val) = each($overview)) {
                print     $val->msgno
                . " - " . $val->date
                . " - " . $val->subject
                . "\n";
        }
```

```
}

imap_close($mbox);
```

imap_fetchbody (PHP 3, PHP 4 >= 4.0.0)

Fetch a particular section of the body of the message

string **imap_fetchbody** (int imap_stream, int msg_number, string part_number [, flags flags]) \linebreak

This function causes a fetch of a particular section of the body of the specified messages as a text string and returns that text string. The section specification is a string of integers delimited by period which index into a body part list as per the IMAP4 specification. Body parts are not decoded by this function.

The options for **imap_fetchbody()** is a bitmask with one or more of the following:

- FT_UID - The `msg_number` is a UID
- FT_PEEK - Do not set the \Seen flag if not already set
- FT_INTERNAL - The return string is in "internal" format, without any attempt to canonicalize CRLF.

See also: imap_fetchstructure().

imap_fetchheader (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns header for a message

string **imap_fetchheader** (int imap_stream, int msgno, int flags) \linebreak

This function causes a fetch of the complete, unfiltered RFC2822 (http://www.faqs.org/rfcs/rfc2822.html) format header of the specified message as a text string and returns that text string.

The options are:

```
FT_UID          The msgno argument is a UID
FT_INTERNAL     The return string is in "internal" format,
                without any attempt to canonicalize to CRLF
                newlines
FT_PREFETCHTEXT The RFC822.TEXT should be pre-fetched at the
                same time.  This avoids an extra RTT on an
                IMAP connection if a full message text is
                desired (e.g. in a "save to local file"
                operation)
```

imap_fetchstructure (PHP 3, PHP 4 >= 4.0.0)

Read the structure of a particular message

object **imap_fetchstructure** (int imap_stream, int msg_number [, int flags]) \linebreak

This function fetches all the structured information for a given message. The optional *flags* parameter only has a single option, *FT_UID*, which tells the function to treat the *msg_number* argument as a *UID*. The returned object includes the envelope, internal date, size, flags and body structure along with a similar object for each mime attachement. The structure of the returned objects is as follows:

Tabella 1. Returned Objects for imap_fetchstructure()

type	Primary body type
encoding	Body transfer encoding
ifsubtype	TRUE if there is a subtype string
subtype	MIME subtype
ifdescription	TRUE if there is a description string
description	Content description string
ifid	TRUE if there is an identification string
id	Identification string
lines	Number of lines
bytes	Number of bytes
ifdisposition	TRUE if there is a disposition string
disposition	Disposition string
ifdparameters	TRUE if the dparameters array exists
dparameters	An array of objects where each object has an "attribute" and a "value" property corresponding to the parameters on the Content-disposition MIMEheader.
ifparameters	TRUE if the parameters array exists
parameters	An array of objects where each object has an "attribute" and a "value" property.
parts	An array of objects identical in structure to the top-level object, each of which corresponds to a MIME body part.

Tabella 2. Primary body type

0	text
1	multipart
2	message
3	application
4	audio
5	image
6	video
7	other

Tabella 3. Transfer encodings

0	7BIT
1	8BIT
2	BINARY
3	BASE64
4	QUOTED-PRINTABLE
5	OTHER

See also: imap_fetchbody().

imap_get_quota (PHP 4 >= 4.0.5)

Retrieve the quota level settings, and usage statics per mailbox

array **imap_get_quota** (int imap_stream, string quota_root) \linebreak

Returns an array with integer values limit and usage for the given mailbox. The value of limit represents the total amount of space allowed for this mailbox. The usage value represents the mailboxes current level of capacity. Will return `FALSE` in the case of failure.

This function is currently only available to users of the c-client2000 library.

imap_stream should be the value returned from an imap_status() call. This stream is required to be opened as the mail admin user for the quota function to work. *quota_root* should normally be in the form of user.name where name is the mailbox you wish to retrieve information about.

Esempio 1. imap_get_quota() example

```
$mbox = imap_open("{your.imap.host}","mailadmin","password",OP_HALFOPEN)
      or die("can’t connect: ".imap_last_error());

$quota_value = imap_get_quota($mbox, "user.kalowsky");
if(is_array($quota_value)) {
```

```
    print "Usage level is: " . $quota_value['usage'];
    print "Limit level is: " . $quota_value['limit'];
}

imap_close($mbox);
```

See also imap_open(), imap_set_quota().

imap_getmailboxes (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Read the list of mailboxes, returning detailed information on each one

array **imap_getmailboxes** (int imap_stream, string ref, string pattern) \linebreak

Returns an array of objects containing mailbox information. Each object has the attributes *name*, specifying the full name of the mailbox; *delimiter*, which is the hierarchy delimiter for the part of the hierarchy this mailbox is in; and *attributes*. *Attributes* is a bitmask that can be tested against:

- LATT_NOINFERIORS - This mailbox has no "children" (there are no mailboxes below this one).
- LATT_NOSELECT - This is only a container, not a mailbox - you cannot open it.
- LATT_MARKED - This mailbox is marked. Only used by UW-IMAPD.
- LATT_UNMARKED - This mailbox is not marked. Only used by UW-IMAPD.

Mailbox names containing international Characters outside the printable ASCII range will be encoded and may be decoded by imap_utf7_decode().

ref should normally be just the server specification as described in imap_open(), and *pattern* specifies where in the mailbox hierarchy to start searching. If you want all mailboxes, pass '*' for *pattern*.

There are two special characters you can pass as part of the *pattern*: '*' and '%'. '*' means to return all mailboxes. If you pass *pattern* as '*', you will get a list of the entire mailbox hierarchy. '%' means to return the current level only. '%' as the *pattern* parameter will return only the top level mailboxes; '~/mail/%' on UW_IMAPD will return every mailbox in the ~/mail directory, but none in subfolders of that directory.

Esempio 1. imap_getmailboxes() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$list = imap_getmailboxes($mbox,"{your.imap.host}","*");
if(is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list))
  {
```

```
    print "($key) ";
    print imap_utf7_decode($val->name).",";
    print "'".$val->delimiter."',";
    print $val->attributes."<br>\n";
  }
} else
  print "imap_getmailboxes failed: ".imap_last_error()."\n";

imap_close($mbox);
```

See also imap_getsubscribed().

imap_getsubscribed (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

List all the subscribed mailboxes

array **imap_getsubscribed** (int imap_stream, string ref, string pattern) \linebreak

This function is identical to imap_getmailboxes(), except that it only returns mailboxes that the user is subscribed to.

imap_header (PHP 3, PHP 4 >= 4.0.0)

Read the header of the message

object **imap_header** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]]) \linebreak

This is an alias to imap_headerinfo() and is identical to this in any way.

imap_headerinfo (PHP 3, PHP 4 >= 4.0.0)

Read the header of the message

object **imap_headerinfo** (int imap_stream, int msg_number [, int fromlength [, int subjectlength [, string defaulthost]]]) \linebreak

This function returns an object of various header elements.

remail, date, Date, subject, Subject, in_reply_to, message_id,
newsgroups, followup_to, references

message flags:
Recent - 'R' if recent and seen,

'N' if recent and not seen,
' ' if not recent
Unseen - 'U' if not seen AND not recent,
' ' if seen OR not seen and recent
Answered -'A' if answered,
' ' if unanswered
Deleted - 'D' if deleted,
' ' if not deleted
Draft - 'X' if draft,
' ' if not draft
Flagged - 'F' if flagged,
' ' if not flagged

NOTE that the Recent/Unseen behavior is a little odd. If you want to know if a message is Unseen, you must check for

Unseen == 'U' || Recent == 'N'

toaddress (full to: line, up to 1024 characters)

to[] (returns an array of objects from the To line, containing):
personal
adl
mailbox
host

fromaddress (full from: line, up to 1024 characters)

from[] (returns an array of objects from the From line, containing):
personal
adl
mailbox
host

ccaddress (full cc: line, up to 1024 characters)
cc[] (returns an array of objects from the Cc line, containing):
personal
adl
mailbox
host

bccaddress (full bcc line, up to 1024 characters)
bcc[] (returns an array of objects from the Bcc line, containing):
personal
adl
mailbox
host

reply_toaddress (full reply_to: line, up to 1024 characters)
reply_to[] (returns an array of objects from the Reply_to line, containing):
personal

adl
mailbox
host

senderaddress (full sender: line, up to 1024 characters)
sender[] (returns an array of objects from the sender line, containing):
personal
adl
mailbox
host

return_path (full return-path: line, up to 1024 characters)
return_path[] (returns an array of objects from the return_path line, containing):
personal
adl
mailbox
host

udate (mail message date in unix time)

fetchfrom (from line formatted to fit *fromlength* characters)

fetchsubject (subject line formatted to fit *subjectlength* characters)

imap_headers (PHP 3, PHP 4 >= 4.0.0)

Returns headers for all messages in a mailbox

array **imap_headers** (int imap_stream) \linebreak

Returns an array of string formatted with header info. One element per mail message.

imap_last_error (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

This function returns the last IMAP error (if any) that occurred during this page request

string **imap_last_error** (void) \linebreak

This function returns the full text of the last IMAP error message that occurred on the current page. The error stack is untouched; calling **imap_last_error()** subsequently, with no intervening errors, will return the same error.

imap_listmailbox (PHP 3, PHP 4 >= 4.0.0)

Read the list of mailboxes

array **imap_listmailbox** (int imap_stream, string ref, string pattern) \linebreak

Returns an array containing the names of the mailboxes. See imap_getmailboxes() for a description of *ref* and *pattern*.

Esempio 1. imap_listmailbox() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$list = imap_listmailbox($mbox,"{your.imap.host}","*");
if(is_array($list)) {
  reset($list);
  while (list($key, $val) = each($list))
    print imap_utf7_decode($val)."<br>\n";
} else
  print "imap_listmailbox failed: ".imap_last_error()."\n";

imap_close($mbox);
```

imap_listsubscribed (PHP 3, PHP 4 >= 4.0.0)

List all the subscribed mailboxes

array **imap_listsubscribed** (int imap_stream, string ref, string pattern) \linebreak

Returns an array of all the mailboxes that you have subscribed. This is almost identical to imap_listmailbox(), but will only return mailboxes the user you logged in as has subscribed.

imap_mail (PHP 3>= 3.0.14, PHP 4 >= 4.0.0)

Send an email message

string **imap_mail** (string to, string subject, string message [, string additional_headers [, string cc [, string bcc [, string rpath]]]]) \linebreak

This function allows sending of emails with correct handling of Cc and Bcc receivers. The parameters to, cc and bcc are all strings and are all parsed as rfc822 address lists. The receivers specified in bcc will get the mail, but are excluded from the headers. Use the rpath parameter to specify return path. This is useful when using php as a mail client for multiple users.

imap_mail_compose (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Create a MIME message based on given envelope and body sections

string **imap_mail_compose** (array envelope, array body) \linebreak

Esempio 1. imap_mail_compose() example

```
<?php

$envelope["from"]="musone@afterfive.com";
$envelope["to"]="musone@darkstar";
$envelope["cc"]="musone@edgeglobal.com";

$part1["type"]=TYPEMULTIPART;
$part1["subtype"]="mixed";

$filename="/tmp/imap.c.gz";
$fp=fopen($filename,"r");
$contents=fread($fp,filesize($filename));
fclose($fp);

$part2["type"]=TYPEAPPLICATION;
$part2["encoding"]=ENCBINARY;
$part2["subtype"]="octet-stream";
$part2["description"]=basename($filename);
$part2["contents.data"]=$contents;

$part3["type"]=TYPETEXT;
$part3["subtype"]="plain";
$part3["description"]="description3";
$part3["contents.data"]="contents.data3\n\n\n\t";

$body[1]=$part1;
$body[2]=$part2;
$body[3]=$part3;

echo nl2br(imap_mail_compose($envelope,$body));

?>
```

imap_mail_copy (PHP 3, PHP 4 >= 4.0.0)

Copy specified messages to a mailbox

int **imap_mail_copy** (int imap_stream, string msglist, string mbox [, int flags]) \linebreak

Returns `TRUE` on success and `FALSE` on error.

Copies mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers (as described in RFC2060 (http://www.faqs.org/rfcs/rfc2060.html)).

Flags is a bitmask of one or more of

- CP_UID - the sequence numbers contain UIDS
- CP_MOVE - Delete the messages from the current mailbox after copying

imap_mail_move (PHP 3, PHP 4 >= 4.0.0)

Move specified messages to a mailbox

int **imap_mail_move** (int imap_stream, string msglist, string mbox [, int flags]) \linebreak

Moves mail messages specified by *msglist* to specified mailbox. *msglist* is a range not just message numbers (as described in RFC2060 (http://www.faqs.org/rfcs/rfc2060.html)).

Flags is a bitmask and may contain the single option

- CP_UID - the sequence numbers contain UIDS

Returns TRUE on success and FALSE on error.

imap_mailboxmsginfo (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Get information about the current mailbox

object **imap_mailboxmsginfo** (int imap_stream) \linebreak

Returns information about the current mailbox. Returns FALSE on failure.

The **imap_mailboxmsginfo()** function checks the current mailbox status on the server. It is similar to imap_status(), but will additionally sum up the size of all messages in the mailbox, which will take some additional time to execute. It returns the information in an object with following properties.

Tabella 1. Mailbox properties

Date	date of last change
Driver	driver
Mailbox	name of the mailbox
Nmsgs	number of messages
Recent	number of recent messages
Unread	number of unread messages
Deleted	number of deleted messages
Size	mailbox size

Esempio 1. imap_mailboxmsginfo() example

```
<?php

$mbox = imap_open("{your.imap.host}INBOX","username", "password")
      or die("can't connect: ".imap_last_error());

$check = imap_mailboxmsginfo($mbox);

if($check) {
    print "Date: "    . $check->Date    ."<br>\n" ;
    print "Driver: "  . $check->Driver  ."<br>\n" ;
    print "Mailbox: " . $check->Mailbox ."<br>\n" ;
    print "Messages: ". $check->Nmsgs   ."<br>\n" ;
    print "Recent: "  . $check->Recent  ."<br>\n" ;
    print "Unread: "  . $check->Unread  ."<br>\n" ;
    print "Deleted: " . $check->Deleted ."<br>\n" ;
    print "Size: "    . $check->Size    ."<br>\n" ;
} else {
    print "imap_check() failed: ".imap_last_error(). "<br>\n";
}

imap_close($mbox);

?>
```

imap_mime_header_decode (PHP 3>= 3.0.17, PHP 4 >= 4.0.0)

Decode MIME header elements

array **imap_mime_header_decode** (string text) \linebreak

imap_mime_header_decode() function decodes MIME message header extensions that are non ASCII text (see RFC2047 (http://www.faqs.org/rfcs/rfc2047.html)) The decoded elements are returned in an array of objects, where each object has two properties, "charset" & "text". If the element hasn't been encoded, and in other words is in plain US-ASCII,the "charset" property of that element is set to "default".

Esempio 1. imap_mime_header_decode() example

```
$text="=?ISO-8859-1?Q?Keld_J=F8rn_Simonsen?= <keld@dkuug.dk>";

$elements=imap_mime_header_decode($text);
for($i=0;$i<count($elements);$i++) {
    echo "Charset: {$elements[$i]->charset}\n";
```

```
    echo "Text: {$elements[$i]->text}\n\n";
}
```

In the above example we would have two elements, whereas the first element had previously been encoded with ISO-8859-1, and the second element would be plain US-ASCII.

imap_msgno (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

This function returns the message sequence number for the given UID

int **imap_msgno** (int imap_stream, int uid) \linebreak

This function returns the message sequence number for the given UID. It is the inverse of imap_uid().

imap_num_msg (PHP 3, PHP 4 >= 4.0.0)

Gives the number of messages in the current mailbox

int **imap_num_msg** (int imap_stream) \linebreak

Return the number of messages in the current mailbox.

See also: imap_num_recent() and imap_status().

imap_num_recent (PHP 3, PHP 4 >= 4.0.0)

Gives the number of recent messages in current mailbox

int **imap_num_recent** (int imap_stream) \linebreak

Returns the number of recent messages in the current mailbox.

See also: imap_num_msg() and imap_status().

imap_open (PHP 3, PHP 4 >= 4.0.0)

Open an IMAP stream to a mailbox

int **imap_open** (string mailbox, string username, string password [, int flags]) \linebreak

Returns an IMAP stream on success and FALSE on error. This function can also be used to open streams to POP3 and NNTP servers, but some functions and features are only available on IMAP servers.

A mailbox name consists of a server part and a mailbox path on this server. The special name INBOX stands for the current users personal mailbox. The server part, which is enclosed in ’{’ and ’}’, consists of the servers name or ip address, an optional port (prefixed by ’:’), and an optional protocol specification (prefixed by ’/’). The server part is mandatory in all mailbox parameters. Mailbox names that contain international characters besides those in the printable ASCII space have to be encoded with imap_utf7_encode().

The options are a bit mask with one or more of the following:

- OP_READONLY - Open mailbox read-only
- OP_ANONYMOUS - Dont use or update a `.newsrc` for news (NNTP only)
- OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox
- CL_EXPUNGE - Expunge mailbox automatically upon mailbox close

To connect to an IMAP server running on port 143 on the local machine, do the following:

```
$mbox = imap_open ("{localhost:143}INBOX", "user_id", "password");
```

To connect to a POP3 server on port 110 on the local server, use:

```
$mbox = imap_open ("{localhost:110/pop3}INBOX", "user_id", "password");
```

To connect to an SSL IMAP or POP3 server, add /ssl after the protocol specification:

```
$mbox = imap_open ("{localhost:993/imap/ssl}INBOX", "user_id", "password");
```

To connect to an SSL IMAP or POP3 server with a self-signed certificate, add /ssl/novalidate-cert after the protocol specification:

```
$mbox = imap_open ("{localhost:995/pop3/ssl/novalidate-cert}", "user_id", "password");
```

To connect to an NNTP server on port 119 on the local server, use:

```
$nntp = imap_open ("{localhost:119/nntp}comp.test", "", "");
```

To connect to a remote server replace "localhost" with the name or the IP address of the server you want to connect to.

Esempio 1. imap_open() example

```
$mbox = imap_open ("{your.imap.host:143}", "username", "password");

echo "<p><h1>Mailboxes</h1>\n";
$folders = imap_listmailbox ($mbox, "{your.imap.host:143}", "*");

if ($folders == false) {
    echo "Call failed<br>\n";
} else {
    while (list ($key, $val) = each ($folders)) {
        echo $val."<br>\n";
    }
}

echo "<p><h1>Headers in INBOX</h1>\n";
$headers = imap_headers ($mbox);

if ($headers == false) {
    echo "Call failed<br>\n";
} else {
    while (list ($key,$val) = each ($headers)) {
        echo $val."<br>\n";
    }
}

imap_close($mbox);
```

imap_ping (PHP 3, PHP 4 >= 4.0.0)

Check if the IMAP stream is still active

int **imap_ping** (int imap_stream) \linebreak

Returns `TRUE` if the stream is still alive, `FALSE` otherwise.

imap_ping() function pings the stream to see it is still active. It may discover new mail; this is the preferred method for a periodic "new mail check" as well as a "keep alive" for servers which have inactivity timeout. (As PHP scripts do not tend to run that long, i can hardly imagine that this function will be usefull to anyone.)

imap_popen (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Open a persistant IMAP stream to a mailbox

int **imap_popen** (string mailbox, string user, string password [, int options]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imap_qprint (PHP 3, PHP 4 >= 4.0.0)

Convert a quoted-printable string to an 8 bit string

string **imap_qprint** (string string) \linebreak

Convert a quoted-printable string to an 8 bit string (according to RFC2045 (http://www.faqs.org/rfcs/rfc2045.html), section 6.7).

Returns an 8 bit (binary) string.

See also imap_8bit().

imap_renamemailbox (PHP 3, PHP 4 >= 4.0.0)

Rename an old mailbox to new mailbox

int **imap_renamemailbox** (int imap_stream, string old_mbox, string new_mbox) \linebreak

This function renames on old mailbox to new mailbox (see imap_open() for the format of *mbox* names).

Returns TRUE on success and FALSE on error.

See also imap_createmailbox(), imap_deletemailbox(), and imap_open() for the format of *mbox*.

imap_reopen (PHP 3, PHP 4 >= 4.0.0)

Reopen IMAP stream to new mailbox

int **imap_reopen** (int imap_stream, string mailbox [, string flags]) \linebreak

This function reopens the specified stream to a new mailbox on an IMAP or NNTP server.

The options are a bit mask with one or more of the following:

- OP_READONLY - Open mailbox read-only
- OP_ANONYMOUS - Dont use or update a `.newsrc` for news (NNTP only)
- OP_HALFOPEN - For IMAP and NNTP names, open a connection but dont open a mailbox.
- CL_EXPUNGE - Expunge mailbox automatically upon mailbox close (see also imap_delete() and imap_expunge())

Returns `TRUE` on success and `FALSE` on error.

imap_rfc822_parse_adrlist (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Parses an address string

array **imap_rfc822_parse_adrlist** (string address, string default_host) \linebreak

This function parses the address string as defined in RFC2822 (http://www.faqs.org/rfcs/rfc2822.html) and for each address, returns an array of objects. The objects properties are:

- mailbox - the mailbox name (username)
- host - the host name
- personal - the personal name
- adl - at domain source route

Esempio 1. imap_rfc822_parse_adrlist() example

```
$address_string = "Hartmut Holzgraefe <hartmut@cvs.php.net>, postmaster@somedomain.net, r
$address_array  = imap_rfc822_parse_adrlist($address_string,"somedomain.net");
if(! is_array($address_array)) die("somethings wrong\n");

reset($address_array);
while(list($key,$val)=each($address_array)){
  print "mailbox : ".$val->mailbox."<br>\n";
  print "host    : ".$val->host."<br>\n";
  print "personal: ".$val->personal."<br>\n";
  print "adl     : ".$val->adl."<p>\n";
}
```

imap_rfc822_parse_headers (PHP 4 >= 4.0.0)

Parse mail headers from a string

object **imap_rfc822_parse_headers** (string headers [, string defaulthost]) \linebreak

This function returns an object of various header elements, similar to imap_header(), except without the flags and other elements that come from the IMAP server.

imap_rfc822_write_address (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Returns a properly formatted email address given the mailbox, host, and personal info.

string **imap_rfc822_write_address** (string mailbox, string host, string personal) \linebreak

Returns a properly formatted email address as defined in RFC2822 (http://www.faqs.org/rfcs/rfc2822.html) given the mailbox, host, and personal info.

Esempio 1. imap_rfc822_write_address() example

```
print imap_rfc822_write_address("hartmut","cvs.php.net","Hartmut Holzgraefe")."\n";
```

imap_scanmailbox (PHP 3, PHP 4 >= 4.0.0)

Read the list of mailboxes, takes a string to search for in the text of the mailbox

array **imap_scanmailbox** (int imap_stream, string ref, string pattern, string content) \linebreak

Returns an array containing the names of the mailboxes that have `content` in the text of the mailbox. This function is similar to imap_listmailbox(), but it will additionally check for the presence of the string `content` inside the mailbox data. See imap_getmailboxes() for a description of `ref` and `pattern`.

imap_search (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

This function returns an array of messages matching the given search criteria

array **imap_search** (int imap_stream, string criteria, int flags) \linebreak

This function performs a search on the mailbox currently opened in the given imap stream. `criteria` is a string, delimited by spaces, in which the following keywords are allowed. Any multi-word arguments (eg. FROM "joey smith") must be quoted.

- ALL - return all messages matching the rest of the criteria
- ANSWERED - match messages with the \\ANSWERED flag set
- BCC "string" - match messages with "string" in the Bcc: field
- BEFORE "date" - match messages with Date: before "date"
- BODY "string" - match messages with "string" in the body of the message
- CC "string" - match messages with "string" in the Cc: field
- DELETED - match deleted messages
- FLAGGED - match messages with the \\FLAGGED (sometimes referred to as Important or Urgent) flag set
- FROM "string" - match messages with "string" in the From: field
- KEYWORD "string" - match messages with "string" as a keyword
- NEW - match new messages
- OLD - match old messages
- ON "date" - match messages with Date: matching "date"
- RECENT - match messages with the \\RECENT flag set
- SEEN - match messages that have been read (the \\SEEN flag is set)
- SINCE "date" - match messages with Date: after "date"
- SUBJECT "string" - match messages with "string" in the Subject:
- TEXT "string" - match messages with text "string"
- TO "string" - match messages with "string" in the To:
- UNANSWERED - match messages that have not been answered
- UNDELETED - match messages that are not deleted
- UNFLAGGED - match messages that are not flagged
- UNKEYWORD "string" - match messages that do not have the keyword "string"
- UNSEEN - match messages which have not been read yet

For example, to match all unanswered messages sent by Mom, you'd use: "UNANSWERED FROM mom". Searches appear to be case insensitive. This list of criteria is from a reading of the UW c-client source code and may be uncomplete or inaccurate (see also RFC2060, section 6.4.4).

Valid values for flags are SE_UID, which causes the returned array to contain UIDs instead of messages sequence numbers.

imap_set_quota (PHP 4 >= 4.0.5)

Sets a quota for a given mailbox

int **imap_set_quota** (int imap_stream, string quota_root, int quota_limit) \linebreak

Sets an upper limit quota on a per mailbox basis. This function requires the `imap_stream` to have been opened as the mail administrator account. It will not work if opened as any other user.

This function is currently only available to users of the c-client2000 library.

imap_stream is the stream pointer returned from a imap_open() call. This stream must be opened as the mail administrator, other wise this function will fail. *quota_root* is the mailbox to have a quota set. This should follow the IMAP standard format for a mailbox, 'user.name'. *quota_limit* is the maximum size (in KB) for the *quota_root*.

Returns TRUE on success and FALSE on error.

Esempio 1. imap_set_quota() example

```
$mbox = imap_open ("{your.imap.host:143}", "mailadmin", "password");

if(!imap_set_quota($mbox, "user.kalowsky", 3000)) {
    print "Error in setting quota\n";
    return;
}

imap_close($mbox);
```

See also imap_open(), **imap_set_quota()**.

imap_setacl (PHP 4 >= 4.1.0)

Sets the ACL for a giving mailbox

int **imap_setacl** (int stream_id, string mailbox, string id, string rights) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imap_setflag_full (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Sets flags on messages

string **imap_setflag_full** (int stream, string sequence, string flag, string options) \linebreak

This function causes a store to add the specified flag to the flags set for the messages in the specified sequence.

The flags which you can set are "\\Seen", "\\Answered", "\\Flagged", "\\Deleted", "\\Draft", and "\\Recent" (as defined by RFC2060).

The options are a bit mask with one or more of the following:

ST_UID The sequence argument contains UIDs instead of sequence numbers

Esempio 1. imap_setflag_full() example

```
$mbox = imap_open("{your.imap.host:143}","username","password")
     or die("can't connect: ".imap_last_error());

$status = imap_setflag_full($mbox,"2,5","\\Seen \\Flagged");

print gettype($status)."\n";
print $status."\n";

imap_close($mbox);
```

imap_sort (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Sort an array of message headers

array **imap_sort** (int stream, int criteria, int reverse, int options) \linebreak

Returns an array of message numbers sorted by the given parameters.

Reverse is 1 for reverse-sorting.

Criteria can be one (and only one) of the following:

SORTDATE message Date
SORTARRIVAL arrival date
SORTFROM mailbox in first From address
SORTSUBJECT message Subject
SORTTO mailbox in first To address
SORTCC mailbox in first cc address
SORTSIZE size of message in octets

The flags are a bitmask of one or more of the following:

SE_UID Return UIDs instead of sequence numbers
SE_NOPREFETCH Don't prefetch searched messages.

imap_status (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

This function returns status information on a mailbox other than the current one

object **imap_status** (int imap_stream, string mailbox, int options) \linebreak

This function returns an object containing status information. Valid flags are:

- SA_MESSAGES - set status->messages to the number of messages in the mailbox
- SA_RECENT - set status->recent to the number of recent messages in the mailbox
- SA_UNSEEN - set status->unseen to the number of unseen (new) messages in the mailbox
- SA_UIDNEXT - set status->uidnext to the next uid to be used in the mailbox
- SA_UIDVALIDITY - set status->uidvalidity to a constant that changes when uids for the mailbox may no longer be valid
- SA_ALL - set all of the above

status->flags is also set, which contains a bitmask which can be checked against any of the above constants.

Esempio 1. imap_status() example

```
$mbox = imap_open("{your.imap.host}","username","password",OP_HALFOPEN)
      or die("can't connect: ".imap_last_error());

$status = imap_status($mbox,"{your.imap.host}INBOX",SA_ALL);
if($status) {
  print("Messages:   ". $status->messages   )."<br>\n";
  print("Recent:     ". $status->recent     )."<br>\n";
  print("Unseen:     ". $status->unseen     )."<br>\n";
  print("UIDnext:    ". $status->uidnext    )."<br>\n";
  print("UIDvalidity:". $status->uidvalidity)."<br>\n";
} else
  print "imap_status failed: ".imap_last_error()."\n";

imap_close($mbox);
```

imap_subscribe (PHP 3, PHP 4 >= 4.0.0)

Subscribe to a mailbox

int **imap_subscribe** (int imap_stream, string mbox) \linebreak

Subscribe to a new mailbox.

Returns TRUE on success and FALSE on error.

imap_thread (PHP 4 >= 4.1.0)

Return threaded by REFERENCES tree

int **imap_thread** (int stream_id [, int flags]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

imap_uid (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

This function returns the UID for the given message sequence number

int **imap_uid** (int imap_stream, int msgno) \linebreak

This function returns the UID for the given message sequence number. An UID is an unique identifier that will not change over time while a message sequence number may change whenever the content of the mailbox changes. This function is the inverse of imap_msgno().

> **Nota:** This is not supported by POP3 mailboxes.

imap_undelete (PHP 3, PHP 4 >= 4.0.0)

Unmark the message which is marked deleted

int **imap_undelete** (int imap_stream, int msg_number) \linebreak

This function removes the deletion flag for a specified message, which is set by imap_delete() or imap_mail_move().

Returns TRUE on success and FALSE on error.

imap_unsubscribe (PHP 3, PHP 4 >= 4.0.0)

Unsubscribe from a mailbox

int **imap_unsubscribe** (int imap_stream, string mbox) \linebreak

Unsubscribe from a specified mailbox.

Returns TRUE on success and FALSE on error.

imap_utf7_decode (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Decodes a modified UTF-7 encoded string.

string **imap_utf7_decode** (string text) \linebreak

Decodes modified UTF-7 *text* into 8bit data.

Returns the decoded 8bit data, or FALSE if the input string was not valid modified UTF-7. This function is needed to decode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (original UTF-7 was defned in RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

imap_utf7_encode (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Converts 8bit data to modified UTF-7 text.

string **imap_utf7_encode** (string data) \linebreak

Converts 8bit *data* to modified UTF-7 text. This is needed to encode mailbox names that contain international characters outside of the printable ASCII range. The modified UTF-7 encoding is defined in RFC 2060 (http://www.faqs.org/rfcs/rfc2060.html), section 5.1.3 (original UTF-7 was defned in RFC1642 (http://www.faqs.org/rfcs/rfc1642.html)).

Returns the modified UTF-7 text.

imap_utf8 (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Converts text to UTF8

string **imap_utf8** (string text) \linebreak

Converts the given *text* to UTF8 (as defined in RFC2044 (http://www.faqs.org/rfcs/rfc2044.html)).

XLIV. Informix functions

The Informix driver for Informix (IDS) 7.x, SE 7.x, Universal Server (IUS) 9.x and IDS 2000 is implemented in "ifx.ec" and "php3_ifx.h" in the informix extension directory. IDS 7.x support is fairly complete, with full support for BYTE and TEXT columns. IUS 9.x support is partly finished: the new data types are there, but SLOB and CLOB support is still under construction.

Configuration notes: You need a version of ESQL/C to compile the PHP Informix driver. ESQL/C versions from 7.2x on should be OK. ESQL/C is now part of the Informix Client SDK.

Make sure that the "INFORMIXDIR" variable has been set, and that $INFORMIXDIR/bin is in your PATH before you run the "configure" script.

The configure script will autodetect the libraries and include directories, if you run "configure --with_informix=yes". You can override this detection by specifying "IFX_LIBDIR", "IFX_LIBS" and "IFX_INCDIR" in the environment. The configure script will also try to detect your Informix server version. It will set the "HAVE_IFX_IUS" conditional compilation variable if your Informix version >= 9.00.

Runtime considerations: Make sure that the Informix environment variables INFORMIXDIR and INFORMIXSERVER are available to the PHP ifx driver, and that the INFORMIX bin directory is in the PATH. Check this by running a script that contains a call to phpinfo() before you start testing. The phpinfo() output should list these environment variables. This is TRUE for both CGI php and Apache mod_php. You may have to set these environment variables in your Apache startup script.

The Informix shared libraries should also be available to the loader (check LD_LINBRARY_PATH or ld.so.conf/ldconfig).

Some notes on the use of BLOBs (TEXT and BYTE columns): BLOBs are normally addressed by BLOB identifiers. Select queries return a "blob id" for every BYTE and TEXT column. You can get at the contents with "string_var = ifx_get_blob($blob_id);" if you choose to get the BLOBs in memory (with : "ifx_blobinfile(0);"). If you prefer to receive the content of BLOB columns in a file, use "ifx_blobinfile(1);", and "ifx_get_blob($blob_id);" will get you the filename. Use normal file I/O to get at the blob contents.

For insert/update queries you must create these "blob id's" yourself with "ifx_create_blob();". You then plug the blob id's into an array, and replace the blob columns with a question mark (?) in the query string. For updates/inserts, you are responsible for setting the blob contents with ifx_update_blob().

The behaviour of BLOB columns can be altered by configuration variables that also can be set at runtime :

configuration variable : ifx.textasvarchar

configuration variable : ifx.byteasvarchar

runtime functions :

ifx_textasvarchar(0) : use blob id's for select queries with TEXT columns

ifx_byteasvarchar(0) : use blob id's for select queries with BYTE columns

ifx_textasvarchar(1) : return TEXT columns as if they were VARCHAR columns, so that you don't need to use blob id's for select queries.

ifx_byteasvarchar(1) : return BYTE columns as if they were VARCHAR columns, so that you

don't need to use blob id's for select queries.

configuration variable : ifx.blobinfile

runtime function :

ifx_blobinfile_mode(0) : return BYTE columns in memory, the blob id lets you get at the contents.

ifx_blobinfile_mode(1) : return BYTE columns in a file, the blob id lets you get at the file name.

If you set ifx_text/byteasvarchar to 1, you can use TEXT and BYTE columns in select queries just like normal (but rather long) VARCHAR fields. Since all strings are "counted" in PHP, this remains "binary safe". It is up to you to handle this correctly. The returned data can contain anything, you are responsible for the contents.

If you set ifx_blobinfile to 1, use the file name returned by ifx_get_blob(..) to get at the blob contents. Note that in this case YOU ARE RESPONSIBLE FOR DELETING THE TEMPORARY FILES CREATED BY INFORMIX when fetching the row. Every new row fetched will create new temporary files for every BYTE column.

The location of the temporary files can be influenced by the environment variable "blobdir", default is "." (the current directory). Something like : putenv(blobdir=tmpblob"); will ease the cleaning up of temp files accidentally left behind (their names all start with "blb").

Automatically trimming "char" (SQLCHAR and SQLNCHAR) data: This can be set with the configuration variable

ifx.charasvarchar : if set to 1 trailing spaces will be automatically trimmed, to save you some "chopping".

`NULL` values: The configuration variable ifx.nullformat (and the runtime function ifx_nullformat()) when set to `TRUE` will return `NULL` columns as the string "`NULL`", when set to `FALSE` they return the empty string. This allows you to discriminate between `NULL` columns and empty columns.

ifx_affected_rows (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Get number of rows affected by a query

int **ifx_affected_rows** (int result_id) \linebreak

result_id is a valid result id returned by ifx_query() or ifx_prepare().

Returns the number of rows affected by a query associated with *result_id*.

For inserts, updates and deletes the number is the real number (sqlerrd[2]) of affected rows. For selects it is an estimate (sqlerrd[0]). Don't rely on it. The database server can never return the actual number of rows that will be returned by a SELECT because it has not even begun fetching them at this stage (just after the "PREPARE" when the optimizer has determined the query plan).

Useful after ifx_prepare() to limit queries to reasonable result sets.

See also: ifx_num_rows()

Esempio 1. Informix affected rows

```
$rid = ifx_prepare ("select * from emp
                     where name like " . $name, $connid);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
```

ifx_blobinfile_mode (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Set the default blob mode for all select queries

void **ifx_blobinfile_mode** (int mode) \linebreak

Set the default blob mode for all select queries. Mode "0" means save Byte-Blobs in memory, and mode "1" means save Byte-Blobs in a file.

ifx_byteasvarchar (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Set the default byte mode

void **ifx_byteasvarchar** (int mode) \linebreak

Sets the default byte mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_close (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Close Informix connection

int **ifx_close** ([int link_identifier]) \linebreak

Returns: always TRUE.

ifx_close() closes the link to an Informix database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

ifx_close() will not close persistent links generated by ifx_pconnect().

See also: ifx_connect(), and ifx_pconnect().

Esempio 1. Closing a Informix connection

```
$conn_id = ifx_connect ("mydb@ol_srv", "itsme", "mypassword");
... some queries and stuff ...
ifx_close($conn_id);
```

ifx_connect (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Open Informix server connection

int **ifx_connect** ([string database [, string userid [, string password]]]) \linebreak

Returns a connection identifier on success, or FALSE on error.

ifx_connect() establishes a connection to an Informix server. All of the arguments are optional, and if they're missing, defaults are taken from values supplied in configuration file (ifx.default_host for the host (Informix libraries will use INFORMIXSERVER environment value if not defined), ifx.default_user for user, ifx.default_password for the password (none if not defined).

In case a second call is made to **ifx_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling ifx_close().

See also ifx_pconnect(), and ifx_close().

Esempio 1. Connect to a Informix database

```
$conn_id = ifx_connect ("mydb@ol_srv1", "imyself", "mypassword");
```

ifx_copy_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Duplicates the given blob object

int **ifx_copy_blob** (int bid) \linebreak

Duplicates the given blob object. *bid* is the ID of the blob object.

Returns FALSE on error otherwise the new blob object-id.

ifx_create_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Creates an blob object

int **ifx_create_blob** (int type, int mode, string param) \linebreak

Creates an blob object.

type: 1 = TEXT, 0 = BYTE

mode: 0 = blob-object holds the content in memory, 1 = blob-object holds the content in file.

param: if mode = 0: pointer to the content, if mode = 1: pointer to the filestring.

Return FALSE on error, otherwise the new blob object-id.

ifx_create_char (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Creates an char object

int **ifx_create_char** (string param) \linebreak

Creates an char object. *param* should be the char content.

ifx_do (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Execute a previously prepared SQL-statement

int **ifx_do** (int result_id) \linebreak

Returns TRUE on success, FALSE on error.

Executes a previously prepared query or opens a cursor for it.

Does NOT free *result_id* on error.

Also sets the real number of ifx_affected_rows() for non-select statements for retrieval by ifx_affected_rows()

See also: ifx_prepare(). There is a example.

ifx_error (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns error code of last Informix call

string **ifx_error** (void) \linebreak

The Informix error codes (SQLSTATE & SQLCODE) formatted as follows :

x [SQLSTATE = aa bbb SQLCODE=cccc]

where x = space : no error

E : error

N : no more data

W : warning

? : undefined

If the "x" character is anything other than space, SQLSTATE and SQLCODE describe the error in more detail.

See the Informix manual for the description of SQLSTATE and SQLCODE

Returns in a string one character describing the general results of a statement and both SQLSTATE and SQLCODE associated with the most recent SQL statement executed. The format of the string is "(char) [SQLSTATE=(two digits) (three digits) SQLCODE=(one digit)]". The first character can be ' ' (space) (success), 'W' (the statement caused some warning), 'E' (an error happened when executing the statement) or 'N' (the statement didn't return any data).

See also: ifx_errormsg()

ifx_errormsg (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Returns error message of last Informix call

string **ifx_errormsg** ([int errorcode]) \linebreak

Returns the Informix error message associated with the most recent Informix error, or, when the optional "*errorcode*" param is present, the error message corresponding to "*errorcode*".

See also: ifx_error()

```
printf("%s\n&lt;br>", ifx_errormsg(-201));
```

ifx_fetch_row (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Get row as enumerated array

array **ifx_fetch_row** (int result_id [, mixed position]) \linebreak

Returns an associative array that corresponds to the fetched row, or FALSE if there are no more rows.

Blob columns are returned as integer blob id values for use in ifx_get_blob() unless you have used ifx_textasvarchar(1) or ifx_byteasvarchar(1), in which case blobs are returned as string values. Returns FALSE on error

result_id is a valid resultid returned by ifx_query() or ifx_prepare() (select type queries only!).

position is an optional parameter for a "fetch" operation on "scroll" cursors: "NEXT", "PREVIOUS", "CURRENT", "FIRST", "LAST" or a number. If you specify a number, an "absolute" row fetch is executed. This parameter is optional, and only valid for SCROLL cursors.

ifx_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0, with the column name as key.

Subsequent calls to **ifx_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

Esempio 1. Informix fetch rows

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                     $connid, IFX_SCROLL);
if (! $rid) {
    ... error ...
}
$rowcount = ifx_affected_rows($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do ($rid)) {
   ... error ...
}
$row = ifx_fetch_row ($rid, "NEXT");
while (is_array($row)) {
    for(reset($row); $fieldname=key($row); next($row)) {
        $fieldvalue = $row[$fieldname];
        printf ("%s = %s,", $fieldname, $fieldvalue);
    }
    printf("\n<br>");
    $row = ifx_fetch_row ($rid, "NEXT");
}
ifx_free_result ($rid);
```

ifx_fieldproperties (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

List of SQL fieldproperties

array **ifx_fieldproperties** (int result_id) \linebreak

Returns an associative array with fieldnames as key and the SQL fieldproperties as data for a query with *result_id*. Returns FALSE on error.

Returns the Informix SQL fieldproperties of every field in the query as an associative array. Properties are encoded as: "SQLTYPE;length;precision;scale;ISNULLABLE" where SQLTYPE = the Informix type like "SQLVCHAR" etc. and ISNULLABLE = "Y" or "N".

Esempio 1. Informix SQL fieldproperties

```
$properties = ifx_fieldproperties ($resultid);
if (! isset($properties)) {
  ... error ...
}
for ($i = 0; $i < count($properties); $i++) {
    $fname = key ($properties);
    printf ("%s:\t type =  %s\n", $fname, $properties[$fname]);
    next ($properties);
}
```

ifx_fieldtypes (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

List of Informix SQL fields

array **ifx_fieldtypes** (int result_id) \linebreak

Returns an associative array with fieldnames as key and the SQL fieldtypes as data for query with *result_id*. Returns FALSE on error.

Esempio 1. Fieldnames and SQL fieldtypes

```
$types = ifx_fieldtypes ($resultid);
if (! isset ($types)) {
  ... error ...
}
for ($i = 0; $i < count($types); $i++) {
    $fname = key($types);
    printf("%s :\t type =  %s\n", $fname, $types[$fname]);
    next($types);
}
```

ifx_free_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Deletes the blob object

int **ifx_free_blob** (int bid) \linebreak

Deletes the blobobject for the given blob object-id *bid*. Returns FALSE on error otherwise TRUE.

ifx_free_char (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deletes the char object

int **ifx_free_char** (int bid) \linebreak

Deletes the charobject for the given char object-id *bid*. Returns FALSE on error otherwise TRUE.

ifx_free_result (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Releases resources for the query

int **ifx_free_result** (int result_id) \linebreak

Releases resources for the query associated with *result_id*. Returns FALSE on error.

ifx_get_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Return the content of a blob object

int **ifx_get_blob** (int bid) \linebreak

Returns the content of the blob object for the given blob object-id *bid*.

ifx_get_char (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Return the content of the char object

int **ifx_get_char** (int bid) \linebreak

Returns the content of the char object for the given char object-id *bid*.

ifx_getsqlca (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Get the contents of sqlca.sqlerrd[0..5] after a query

array **ifx_getsqlca** (int result_id) \linebreak

result_id is a valid result id returned by ifx_query() or ifx_prepare().

Returns a pseudo-row (associative array) with sqlca.sqlerrd[0] ... sqlca.sqlerrd[5] after the query associated with *result_id*.

For inserts, updates and deletes the values returned are those as set by the server after executing the query. This gives access to the number of affected rows and the serial insert value. For SELECTs the values are those saved after the PREPARE statement. This gives access to the *estimated* number of affected rows. The use of this function saves the overhead of executing a "select dbinfo('sqlca.sqlerrdx')" query, as it retrieves the values that were saved by the ifx driver at the appropriate moment.

Esempio 1. Retrieve Informix sqlca.sqlerrd[x] values

```
/* assume the first column of 'sometable' is a serial */
$qid = ifx_query("insert into sometable
                  values (0, '2nd column', 'another column') ", $connid);
if (! $qid) {
    ... error ...
}
$sqlca = ifx_getsqlca ($qid);
$serial_value = $sqlca["sqlerrd1"];
echo "The serial value of the inserted row is : " . $serial_value<br>\n";
```

ifx_htmltbl_result (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Formats all rows of a query into a HTML table

int **ifx_htmltbl_result** (int result_id [, string html_table_options]) \linebreak

Returns the number of rows fetched or `FALSE` on error.

Formats all rows of the *result_id* query into a html table. The optional second argument is a string of <table> tag options

Esempio 1. Informix results as HTML table

```
$rid = ifx_prepare ("select * from emp where name like " . $name,
                     $connid, IFX_SCROLL);
if (! $rid) {
   ... error ...
}
$rowcount = ifx_affected_rows ($rid);
if ($rowcount > 1000) {
    printf ("Too many rows in result set (%d)\n<br>", $rowcount);
    die ("Please restrict your query<br>\n");
}
if (! ifx_do($rid) {
  ... error ...
```

```
}

ifx_htmltbl_result ($rid, "border=\"2\"");

ifx_free_result($rid);
```

ifx_nullformat (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Sets the default return value on a fetch row

void **ifx_nullformat** (int mode) \linebreak

Sets the default return value of a NULL-value on a fetch row. Mode "0" returns "", and mode "1" returns "`NULL`".

ifx_num_fields (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Returns the number of columns in the query

int **ifx_num_fields** (int result_id) \linebreak

Returns the number of columns in query for *`result_id`* or `FALSE` on error

After preparing or executing a query, this call gives you the number of columns in the query.

ifx_num_rows (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Count the rows already fetched from a query

int **ifx_num_rows** (int result_id) \linebreak

Gives the number of rows fetched so far for a query with *`result_id`* after a ifx_query() or ifx_do() query.

ifx_pconnect (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Open persistent Informix connection

int **ifx_pconnect** ([string database [, string userid [, string password]]]) \linebreak

Returns: A positive Informix persistent link identifier on success, or `FALSE` on error

ifx_pconnect() acts very much like ifx_connect() with two major differences.

This function behaves exactly like ifx_connect() when PHP is not running as an Apache module. First, when connecting, the function would first try to find a (persistent) link that's already open with

the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ifx_close() will not close links established by **ifx_pconnect()**).

This type of links is therefore called ’persistent’.

See also: ifx_connect().

ifx_prepare (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Prepare an SQL-statement for execution

int **ifx_prepare** (string query, int conn_id [, int cursor_def, mixed blobidarray]) \linebreak

Returns a integer *result_id* for use by ifx_do(). Sets *affected_rows* for retrieval by the ifx_affected_rows() function.

Prepares *query* on connection *conn_id*. For "select-type" queries a cursor is declared and opened. The optional *cursor_type* parameter allows you to make this a "scroll" and/or "hold" cursor. It’s a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or’ed together.

For either query type the estimated number of affected rows is saved for retrieval by ifx_affected_rows().

If you have BLOB (BYTE or TEXT) columns in the query, you can add a *blobidarray* parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don’t need to bother with blob id’s.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id’s (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: ifx_do().

ifx_query (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Send Informix query

int **ifx_query** (string query [, int link_identifier [, int cursor_type [, mixed blobidarray]]]) \linebreak

Returns: A positive Informix result identifier on success, or FALSE on error.

A "result_id" resource used by other functions to retrieve the query results. Sets "affected_rows" for retrieval by the ifx_affected_rows() function.

ifx_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if ifx_connect() was called, and use it.

Executes `query` on connection `conn_id`. For "select-type" queries a cursor is declared and opened. The optional `cursor_type` parameter allows you to make this a "scroll" and/or "hold" cursor. It's a bitmask and can be either IFX_SCROLL, IFX_HOLD, or both or'ed together. Non-select queries are "execute immediate". IFX_SCROLL and IFX_HOLD are symbolic constants and as such shouldn't be between quotes. I you omit this parameter the cursor is a normal sequential cursor.

For either query type the number of (estimated or real) affected rows is saved for retrieval by ifx_affected_rows().

If you have BLOB (BYTE or TEXT) columns in an update query, you can add a `blobidarray` parameter containing the corresponding "blob ids", and you should replace those columns with a "?" in the query text.

If the contents of the TEXT (or BYTE) column allow it, you can also use "ifx_textasvarchar(1)" and "ifx_byteasvarchar(1)". This allows you to treat TEXT (or BYTE) columns just as if they were ordinary (but long) VARCHAR columns for select queries, and you don't need to bother with blob id's.

With ifx_textasvarchar(0) or ifx_byteasvarchar(0) (the default situation), select queries will return BLOB columns as blob id's (integer value). You can get the value of the blob as a string or file with the blob functions (see below).

See also: ifx_connect().

Esempio 1. Show all rows of the "orders" table as a html table

```
ifx_textasvarchar(1);      // use "text mode" for blobs
$res_id = ifx_query("select * from orders", $conn_id);
if (! $res_id) {
    printf("Can't select orders : %s\n<br>%s<br>\n", ifx_error());
    ifx_errormsg();
    die;
}
ifx_htmltbl_result($res_id, "border=\"1\"");
ifx_free_result($res_id);
```

Esempio 2. Insert some values into the "catalog" table

```
                   // create blob id's for a byte and text column
$textid = ifx_create_blob(0, 0, "Text column in memory");
$byteid = ifx_create_blob(1, 0, "Byte column in memory");
                      // store blob id's in a blobid array
$blobidarray[] = $textid;
$blobidarray[] = $byteid;
                      // launch query
$query = "insert into catalog (stock_num, manu_code, " .
         "cat_descr,cat_picture) values(1,'HRO',?,?)";
$res_id = ifx_query($query, $conn_id, $blobidarray);
```

```
if (! $res_id) {
  ... error ...
}
                     // free result id
ifx_free_result($res_id);
```

ifx_textasvarchar (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Set the default text mode

void **ifx_textasvarchar** (int mode) \linebreak

Sets the default text mode for all select-queries. Mode "0" will return a blob id, and mode "1" will return a varchar with text content.

ifx_update_blob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Updates the content of the blob object

ifx_update_blob (int bid, string content) \linebreak

Updates the content of the blob object for the given blob object *`bid`*. *`content`* is a string with new data. Returns `FALSE` on error otherwise `TRUE`.

ifx_update_char (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Updates the content of the char object

int **ifx_update_char** (int bid, string content) \linebreak

Updates the content of the char object for the given char object *`bid`*. *`content`* is a string with new data. Returns `FALSE` on error otherwise `TRUE`.

ifxus_close_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Deletes the slob object

int **ifxus_close_slob** (int bid) \linebreak

Deletes the slob object on the given slob object-id *`bid`*. Return `FALSE` on error otherwise `TRUE`.

ifxus_create_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Creates an slob object and opens it

int **ifxus_create_slob** (int mode) \linebreak

Creates an slob object and opens it. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. You can also use constants named IFX_LO_RDONLY, IFX_LO_WRONLY etc. Return FALSE on error otherwise the new slob object-id.

ifxus_free_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Deletes the slob object

int **ifxus_free_slob** (int bid) \linebreak

Deletes the slob object. *bid* is the Id of the slob object. Returns FALSE on error otherwise TRUE.

ifxus_open_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Opens an slob object

int **ifxus_open_slob** (long bid, int mode) \linebreak

Opens an slob object. *bid* should be an existing slob id. Modes: 1 = LO_RDONLY, 2 = LO_WRONLY, 4 = LO_APPEND, 8 = LO_RDWR, 16 = LO_BUFFER, 32 = LO_NOBUFFER -> or-mask. Returns FALSE on error otherwise the new slob object-id.

ifxus_read_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Reads nbytes of the slob object

int **ifxus_read_slob** (long bid, long nbytes) \linebreak

Reads nbytes of the slob object. *bid* is a existing slob id and *nbytes* is the number of bytes read. Return FALSE on error otherwise the string.

ifxus_seek_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Sets the current file or seek position

int **ifxus_seek_slob** (long bid, int mode, long offset) \linebreak

Sets the current file or seek position of an open slob object. *bid* should be an existing slob id. Modes: 0 = LO_SEEK_SET, 1 = LO_SEEK_CUR, 2 = LO_SEEK_END and *offset* is an byte offset. Return FALSE on error otherwise the seek position.

ifxus_tell_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Returns the current file or seek position

int **ifxus_tell_slob** (long bid) \linebreak

Returns the current file or seek position of an open slob object *bid* should be an existing slob id. Return FALSE on error otherwise the seek position.

ifxus_write_slob (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Writes a string into the slob object

int **ifxus_write_slob** (long bid, string content) \linebreak

Writes a string into the slob object. *bid* is a existing slob id and *content* the content to write. Return FALSE on error otherwise bytes written.

XLV. Funzioni InterBase

InterBase è un famoso database prodotto da Borland/Inprise. Maggiori informazioni riguardo InterBase sono disponibili a http://www.interbase.com/. Oh, a proposito, InterBase è da poco entrato nel movimento open source!

Nota: Il supporto completo per InterBase 6 è stato aggiunto in PHP 4.0.

Questo database usa il carattere di singolo apice (') come carattere di escape, un comportamento simile al database Sybase, aggiungere al proprio file `php.ini` la seguente direttiva:

```
magic_quotes_sybase = On
```

ibase_blob_add (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Aggiunge dati in un blob creato

int **ibase_blob_add** (int blob_id, string data) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_blob_cancel (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Cancella la creazione di un blob

int **ibase_blob_cancel** (int blob_id) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_blob_close (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Chiude un blob

int **ibase_blob_close** (int blob_id) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_blob_create (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Crea un blob per aggiungerci dei dati

int **ibase_blob_create** ([int link_identifier]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_blob_echo (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Visualizza il contenuto di un blob sul browser

int **ibase_blob_echo** (string blob_id_str) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_blob_get (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ottiene len byte di dati dal blob aperto

string **ibase_blob_get** (int blob_id, int len) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_blob_import (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Create un blob, copy il file al suo interno e lo chiude

string **ibase_blob_import** ([int link_identifier, int file_id]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_blob_info (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Restituisce la lunghezza del blob e altre informazioni utlili

object **ibase_blob_info** (string blob_id_str) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_blob_open (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Apre un blob per ricavare parte di dati

int **ibase_blob_open** (string blob_id) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ibase_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Chiude una connessione ad un database InterBase

int **ibase_close** ([int connection_id]) \linebreak

Chiude il collegamento ad un database InterBase che è stato associato con un id di connessione restituito da ibase_connect(). Se l'id di connessione viene omesso, viene considerato l'ultimo collegamento che è stato aperto. La transazione predefinita sul collegamento viene "committed", le altre transazioni vengono "rolled back".

ibase_commit (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Esegue il commit di una transazione

int **ibase_commit** ([int link_identifier, int trans_number]) \linebreak

Esegue il commit della transazione *trans_number* che è stata creata con ibase_trans().

ibase_connect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Apre una connessione con un database InterBase

int **ibase_connect** (string database [, string username [, string password [, string charset [, int buffers [, int dialect [, string role]]]]]]) \linebreak

Stabilisce una connessione con un server InterBase. Il parametro *database* deve essere un percorso valido di un file di database sul server dove risiede. Se il server non fosse locale, bisogna aggiungere prima del percorso o 'hostname:' (TCP/IP), o '//hostname/' (NetBEUI) o 'hostname@' (IPX/SPX), in base al protocollo di connessione utilizzato. *username* e *password* possono venire specificati anche con le direttive di configurazione del PHP ibase.default_user e ibase.default_password. *charset* è il set di caratteri predefinito per un database. *buffers* è il numero di buffer di database da allocare per la cache dal lato server. Se è 0 o viene omesso, il server usa il suo valore predefinito. *dialect* seleziona il dialetto SQL predefinito per ogni operazione eseguita all'interno di una connessione, e il suo valore predefinito è il più alto supportato dalle librerie del client.

Nel caso di una seconda chiamata fatta con **ibase_connect()** con gli stessi parametri, non verrà creato alcun nuovo collegamento, bensì, l'identificatore del collegamento già aperto verrà restituito. Il collegamento al server verrà chiuso appena termina l'esecuzione dello script, a meno che non venga chiuso prima esplicitamente chiamando ibase_close().

Esempio 1. Esempio di ibase_connect()

```
<?php
    $dbh = ibase_connect($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query($dbh, $stmt);
    while ($row = ibase_fetch_object($sth)) {
        echo $row->email . "\n";
    }
    ibase_free_result($sth);
    ibase_close($dbh);
?>
```

Nota: *buffers* è stato aggiunto in PHP4RC2.

Nota: *dialect* è stato aggiunto in PHP4RC2. E' funzionante solo con InterBase 6 e versioni successive.

Nota: `role` è stato aggiunto in PHP4RC2. E' funzionante solo con InterBase 5 e versioni successive.

Vedere anche: ibase_pconnect().

ibase_errmsg (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Restituisce messaggi di errore

string **ibase_errmsg** (void) \linebreak

Restituisce una stringa contenente un messaggio di errore.

ibase_execute (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Esegue una query preparata in precedenza

int **ibase_execute** (int query [, int bind_args]) \linebreak

Esegue una query preparata da ibase_prepare(). Ciò è molto più efficace che usare ibase_query() se state ripetendo uno stesso tipo di query molte volte cambiando solo alcuni parametri.

```
<?php
    $updates = array(
        1 => 'Eric',
        5 => 'Filip',
        7 => 'Larry'
    );

    $query = ibase_prepare("UPDATE FOO SET BAR = ? WHERE BAZ = ?");

    while (list($baz, $bar) = each($updates)) {
        ibase_execute($query, $bar, $baz);
    }
?>
```

ibase_fetch_object (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ottiene un oggetto da un database InterBase

object **ibase_fetch_object** (int result_id) \linebreak

Elabora una riga come un pseudo-oggetto da un *result_id* ottenuto o da ibase_query() o da ibase_execute().

```
<php
    $dbh = ibase_connect ($host, $username, $password);
    $stmt = 'SELECT * FROM tblname';
    $sth = ibase_query ($dbh, $stmt);
    while ($row = ibase_fetch_object ($sth)) {
        print $row->email . "\n";
    }
    ibase_close ($dbh);
?>
```

Chiamate successive a **ibase_fetch_object()** restituiscono la successiva riga dai risultati della query o FALSE se non vi sono ulteriori righe.

Vedere anche ibase_fetch_row().

ibase_fetch_row (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Elabora una riga da un database InterBase

array **ibase_fetch_row** (int result_identifier) \linebreak

Restituisce un array che corrisponde alla riga ottenuta o FALSE se non ci sono righe rimanenti.

ibase_fetch_row() prende una riga di dati dai risultati associati allo specificato *result_identifier*. La riga viene restituita come un array. Ogni colonna risultante viene immagazzinata in un offset dell'array, l'offset inizia da 0.

Chiamate successive a **ibase_fetch_row()** restituiscono la successiva riga dai risultati della query o FALSE se non vi sono ulteriori righe.

ibase_field_info (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ottiene informazioni su un campo

array **ibase_field_info** (int result, int field number) \linebreak

Restituisce un array con informazioni relative a un campo dopo che una query select è stata eseguita. L'array ha la forma name, alias, relation, length e type.

```
$rs=ibase_query("SELECT * FROM tablename");
$coln = ibase_num_fields($rs);
for ($i=0; $i < $coln; $i++) {
    $col_info = ibase_field_info($rs, $i);
    echo "name: ".$col_info['name']."\n";
    echo "alias: ".$col_info['alias']."\n";
```

```
echo "relation: ".$col_info['relation']."\n";
echo "length: ".$col_info['length']."\n";
echo "type: ".$col_info['type']."\n";
}
```

ibase_free_query (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Libera la memoria allocata da una query preparata

int **ibase_free_query** (int query) \linebreak

Libera una query preparata da ibase_prepare().

ibase_free_result (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Libera la memoria allocata da un result set

int **ibase_free_result** (int result_identifier) \linebreak

Libera un result set che è stato creato da ibase_query().

ibase_num_fields (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ottiene il numero di campi in un result set

int **ibase_num_fields** (int result_id) \linebreak

Restituisce un integer contenente il numero di campi in un result set.

```
<?php
$dbh = ibase_connect ($host, $username, $password);
$stmt = 'SELECT * FROM tblname';
$sth = ibase_query ($dbh, $stmt);

if (ibase_num_fields($sth) > 0) {
while ($row = ibase_fetch_object ($sth)) {
print $row->email . "\n";
}
} else {
die ("Nessun result è stato trovato per la tua query");
}

ibase_close ($dbh);
?>
```

Vedere anche: ibase_field_info().

ibase_pconnect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Crea una connessione persistente ad un database Interbase

int **ibase_connect** (string database [, string username [, string password [, string charset [, int buffers [, int dialect [, string role]]]]]]) \linebreak

ibase_pconnect() agisce in modo molto simile a ibase_connect() con due differenze principali. Innanzitutto, durante la connessione, la funzione cercherà prima di trovare un collegamento (persistente) che è già stato aperto con gli stessi parametri. Se viene trovato, il suo identificatore verrà restituito al posto di aprire una nuova connessione. In secondo luogo, la connessione al server InterBase non verrà chiusa al termine dell'esecuzione dello script. Invece, il collegamento resterà aperto per un uso futuro (ibase_close() non chiuderà i collegamenti stabiliti da **ibase_pconnect()**). Questo tipo di collegamento è perciò chiamato 'persistente'.

> **Nota:** *buffers* è stato aggiunto in PHP4-RC2.

> **Nota:** *dialect* è stato aggiunto in PHP4-RC2. Funziona soltanto con InterBase 6 e superiori.

> **Nota:** *role* è stato aggiunto in PHP4-RC2. Funziona soltanto con InterBase 5 e superiori.

Vedere anche ibase_connect() per il significato dei parametri passati a questa funzione. Sono esattamente gli stessi.

ibase_prepare (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Prepara una query per un successivo binding dei segnaposto dei parametri ed esecuzione

int **ibase_prepare** ([int link_identifier, string query]) \linebreak

Prepara una query per un successivo binding dei segnaposto dei parametri ed esecuzione (tramite ibase_execute()).

ibase_query (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Esegue una query su di un database InterBase

int **ibase_query** ([int link_identifier, string query [, int bind_args]]) \linebreak

Esegue una query su di un database InterBase. Se la query non ha successo, restituisce `FALSE`. Se ha successo e vi sono riga di risultato (come si ha ad esempio con le query SELECT), restituisce un identificatore di risorsa. Se la query ha avuto successo, ma non ci sono risultati, restituisce `TRUE`. Restutuisce `FALSE` se la query fallisce.

Vedere anche ibase_errmsg(), ibase_fetch_row(), ibase_fetch_object() e ibase_free_result().

ibase_rollback (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Esegue il roll back di una transazione

int **ibase_rollback** ([int link_identifier, int trans_number]) \linebreak

Rolls back transaction *`trans_number`* which was created with ibase_trans().

ibase_timefmt (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Imposta il formato delle colonne timestamp, date e time restituite dalle query

int **ibase_timefmt** (string format [, int columntype]) \linebreak

Imposta il formato delle colonne di tipo timestamp, date o time restituite dalle query. Internamente, le colonne vengono formattate dalla funzione C strftime(), quindi fate riferimento alla sua documentazione riguardo al formato della stringa. *`columntype`* è una delle costanti IBASE_TIMESTAMP, IBASE_DATE e IBASE_TIME. Se omessa, è predefinita a IBASE_TIMESTAMP per motivi di compatibilità con il passato.

```
<?php
// Le colonne di tipo TIME di InterBase vengono restituite nella
// forma ’05 hours 37 minutes’.
ibase_timefmt("%H hours %M minutes", IBASE_TIME);
?>
```

Potete impostare anche valori predefiniti per questi formati con la direttiva di configurazione ibase.timestampformat, ibase.dateformat e ibase.timeformat.

> **Nota:** *`columntype`* è stata aggiunta in PHP 4.0. Ha significato solo con InterBase versione 6 e successive.

> **Nota:** Un’incompatibilità con il passato si è avuta nel PHP 4.0 quando la direttiva di configurazione del PHP ibase.timeformat è stata rinominata in ibase.timestampformat e la direttiva ibase.dateformat e ibase.timeformat sono state aggiunte, così che i loro nomi fossero più simili alle loro funzionalità.

ibase_trans (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Inizia una transazione

int **ibase_trans** ([int trans_args [, int link_identifier]]) \linebreak

Inizia una transazione.

XLVI. Ingres II functions

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

These functions allow you to access Ingres II database servers.

In order to have these functions available, you must compile php with Ingres support by using the `--with-ingres` option. You need the Open API library and header files included with Ingres II. If the II_SYSTEM environment variable isn't correctly set you may have to use `--with-ingres=DIR` to specify your Ingres installation directory.

When using this extension with Apache, if Apache does not start and complains with "PHP Fatal error: Unable to start ingres_ii module in Unknown on line 0" then make sure the environement variable II_SYSTEM is correctly set. Adding "export II_SYSTEM="/home/ingres/II" in the script that starts Apache, just before launching httpd, should be fine.

> **Nota:** If you already used PHP extensions to access other database servers, note that Ingres doesn't allow concurrent queries and/or transaction over one connection, thus you won't find any result or transaction handle in this extension. The result of a query must be treated before sending another query, and a transaction must be commited or rolled back before opening another transaction (which is automaticaly done when sending the first query).

ingres_autocommit (PHP 4 >= 4.0.2)

Switch autocommit on or off

bool **ingres_autocommit** ([resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_autocommit() is called before opening a transaction (before the first call to ingres_query() or just after a call to ingres_rollback() or **ingres_autocommit()**) to switch the "autocommit" mode of the server on or off (when the script begins the autocommit mode is off).

When the autocommit mode is on, every query is automaticaly commited by the server, as if ingres_commit() was called after every call to ingres_query().

See also ingres_query(), ingres_rollback(), and ingres_commit().

ingres_close (PHP 4 >= 4.0.2)

Close an Ingres II database connection

bool **ingres_close** ([resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns `TRUE` on success, or `FALSE` on failure.

ingres_close() closes the connection to the Ingres server that's associated with the specified link. If the *link* parameter isn't specified, the last opened link is used.

ingres_close() isn't usually necessary, as it won't close persistent connections and all non-persistent connections are automatically closed at the end of the script.

See also ingres_connect() and ingres_pconnect().

ingres_commit (PHP 4 >= 4.0.2)

Commit a transaction

bool **ingres_commit** ([resource link]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_commit() commits the currently open transaction, making all changes made to the database permanent.

This closes the transaction. A new one can be open by sending a query with ingres_query().

You can also have the server commit automaticaly after every query by calling ingres_autocommit() before opening the transaction.

See also ingres_query(), ingres_rollback(), and ingres_autocommit().

ingres_connect (PHP 4 >= 4.0.2)

Open a connection to an Ingres II database

resource **ingres_connect** ([string database [, string username [, string password]]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns a Ingres II link resource on success, or `FALSE` on failure.

ingres_connect() opens a connection with the Ingres database designated by *`database`*, which follows the syntax *`[node_id::]dbname[/svr_class]`*.

If some parameters are missing, **ingres_connect()** uses the values in `php.ini` for *`ingres.default_database`*, *`ingres.default_user`*, and *`ingres.default_password`*.

The connection is closed when the script ends or when ingres_close() is called on this link.

All the other ingres functions use the last opened link as a default, so you need to store the returned value only if you use more than one link at a time.

Esempio 1. ingres_connect() example

```
<?php
    $link = ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ($link);
?>
```

Esempio 2. ingres_connect() example using default link

```
<?php
    ingres_connect ("mydb", "user", "pass")
        or die ("Could not connect");
    print ("Connected successfully");
    ingres_close ();
?>
```

See also ingres_pconnect() and ingres_close().

ingres_fetch_array (PHP 4 >= 4.0.2)

Fetch a row of result into an array

array **ingres_fetch_array** ([int result_type [, resource link]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_fetch_array() Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

This function is an extended version of ingres_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column.

```
ingres_query(select t1.f1 as foo t2.f1 as bar from t1, t2);
$result = ingres_fetch_array();
$foo = $result["foo"];
$bar = $result["bar"];
```

`result_type` can be INGRES_NUM for enumerated array, INGRES_ASSOC for associative array, or INGRES_BOTH (default).

Speed-wise, the function is identical to ingres_fetch_object(), and almost as quick as ingres_fetch_row() (the difference is insignificant).

Esempio 1. ingres_fetch_array() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_array()) {
    echo $row["user_id"];  # using associative array
    echo $row["fullname"];
    echo $row[1];          # using enumerated array
    echo $row[2];
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_object(), and ingres_fetch_row().

ingres_fetch_object (PHP 4 >= 4.0.2)

Fetch a row of result into an object.

object **ingres_fetch_object** ([int result_type [, resource link]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_fetch_object() Returns an object that corresponds to the fetched row, or FALSE if there are no more rows.

This function is similar to ingres_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional argument *result_type* is a constant and can take the following values: INGRES_ASSOC, INGRES_NUM, and INGRES_BOTH.

Speed-wise, the function is identical to ingres_fetch_array(), and almost as quick as ingres_fetch_row() (the difference is insignificant).

Esempio 1. ingres_fetch_object() example

```
<?php
ingres_connect ($database, $user, $password);
ingres_query ("select * from table");
```

```
while ($row = ingres_fetch_object()) {
    echo $row->user_id;
    echo $row->fullname;
}
?>
```

See also ingres_query(), ingres_num_fields(), ingres_field_name(), ingres_fetch_array(), and ingres_fetch_row().

ingres_fetch_row (PHP 4 >= 4.0.2)

Fetch a row of result into an enumerated array

array **ingres_fetch_row** ([resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_fetch_row() returns an array that corresponds to the fetched row, or FALSE if there are no more rows. Each result column is stored in an array offset, starting at offset 1.

Subsequent call to **ingres_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

Esempio 1. ingres_fetch_row() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also ingres_num_fields(), ingres_query(), ingres_fetch_array(), and ingres_fetch_object().

ingres_field_length (PHP 4 >= 4.0.2)

Get the length of a field

int **ingres_field_length** (int index [, resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_field_length() returns the length of a field. This is the number of bytes used by the server to store the field. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_field_name (PHP 4 >= 4.0.2)

Get the name of a field in a query result.

string **ingres_field_name** (int index [, resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_field_name() returns the name of a field in a query result, or `FALSE` on failure.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object() and ingres_fetch_row().

ingres_field_nullable (PHP 4 >= 4.0.2)

Test if a field is nullable

bool **ingres_field_nullable** (int index [, resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_field_nullable() returns `TRUE` if the field can be set to the `NULL` value and `FALSE` if it can't.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_field_precision (PHP 4 >= 4.0.2)

Get the precision of a field

int **ingres_field_precision** (int index [, resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_field_precision() returns the precision of a field. This value is used only for decimal, float and money SQL data types. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_field_scale (PHP 4 >= 4.0.2)

Get the scale of a field

int **ingres_field_scale** (int index [, resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_field_scale() returns the scale of a field. This value is used only for the decimal SQL data type. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

index is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_field_type (PHP 4 >= 4.0.2)

Get the type of a field in a query result

string **ingres_field_type** (int index [, resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_field_type() returns the type of a field in a query result, or `FALSE` on failure. Examples of types returned are "IIAPI_BYTE_TYPE", "IIAPI_CHA_TYPE", "IIAPI_DTE_TYPE", "IIAPI_FLT_TYPE", "IIAPI_INT_TYPE", "IIAPI_VCH_TYPE". Some of these types can map to more than one SQL type depending on the length of the field (see ingres_field_length()). For example "IIAPI_FLT_TYPE" can be a float4 or a float8. For detailed information, see the Ingres/OpenAPI User Guide - Appendix C.

`index` is the number of the field and must be between 1 and the value given by ingres_num_fields().

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_num_fields (PHP 4 >= 4.0.2)

Get the number of fields returned by the last query

int **ingres_num_fields** ([resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_num_fields() returns the number of fields in the results returned by the Ingres server after a call to ingres_query()

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_num_rows (PHP 4 >= 4.0.2)

Get the number of rows affected or returned by the last query

int **ingres_num_rows** ([resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

For delete, insert or update queries, **ingres_num_rows()** returns the number of rows affected by the query. For other queries, **ingres_num_rows()** returns the number of rows in the query's result.

> **Nota:** This function is mainly meant to get the number of rows modified in the database. If this function is called before using ingres_fetch_array(), ingres_fetch_object() or ingres_fetch_row() the server will delete the result's data and the script won't be able to get them.
>
> You should instead retrieve the result's data using one of these fetch functions in a loop until it returns FALSE, indicating that no more results are available.

See also ingres_query(), ingres_fetch_array(), ingres_fetch_object(), and ingres_fetch_row().

ingres_pconnect (PHP 4 >= 4.0.2)

Open a persistent connection to an Ingres II database

resource **ingres_pconnect** ([string database [, string username [, string password]]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns a Ingres II link resource on success, or FALSE on failure.

See ingres_connect() for parameters details and examples. There are only 2 differences between **ingres_pconnect()** and ingres_connect() : First, when connecting, the function will first try to find a (persistent) link that's already opened with the same parameters. If one is found, an identifier for it will be returned instead of opening a new connection. Second, the connection to the Ingres server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (ingres_close() will not close links established by **ingres_pconnect()**). This type of link is therefore called 'persistent'.

See also ingres_connect() and ingres_close().

ingres_query (PHP 4 >= 4.0.2)

Send a SQL query to Ingres II

bool **ingres_query** (string query [, resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns `TRUE` on success, or `FALSE` on failure.

ingres_query() sends the given *`query`* to the Ingres server. This query must be a valid SQL query (see the Ingres SQL reference guide)

The query becomes part of the currently open transaction. If there is no open transaction, **ingres_query()** opens a new transaction. To close the transaction, you can either call ingres_commit() to commit the changes made to the database or ingres_rollback() to cancel these changes. When the script ends, any open transaction is rolled back (by calling ingres_rollback()). You can also use ingres_autocommit() before opening a new transaction to have every SQL query immediatly commited.

Some types of SQL queries can't be sent with this function:

- close (see ingres_close())
- commit (see ingres_commit())
- connect (see ingres_connect())
- disconnect (see ingres_close())
- get dbevent
- prepare to commit
- rollback (see ingres_rollback())
- savepoint
- set autocommit (see ingres_autocommit())
- all cursor related queries are unsupported

Esempio 1. ingres_query() example

```
<?php
ingres_connect ($database, $user, $password);

ingres_query ("select * from table");
while ($row = ingres_fetch_row()) {
    echo $row[1];
    echo $row[2];
}
?>
```

See also ingres_fetch_array(), ingres_fetch_object(), ingres_fetch_row(), ingres_commit(), ingres_rollback(), and ingres_autocommit().

ingres_rollback (PHP 4 >= 4.0.2)

Roll back a transaction

bool **ingres_rollback** ([resource link]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ingres_rollback() rolls back the currently open transaction, actually canceling all changes made to the database during the transaction.

This closes the transaction. A new one can be open by sending a query with ingres_query().

See also ingres_query(), ingres_commit(), and ingres_autocommit().

XLVII. IRC Gateway Functions

What is ircg?

With ircg you can build powerful, fast and scalable webchat solutions in conjunction with dedicated or public IRC servers.

Platforms

IRCG runs under

- AIX
- FreeBSD
- HP-UX
- Irix
- Linux
- Solaris
- Tru64

Requirements

To install and use IRCG you need the following software:

1. IRCG-Library (http://www.schumann.cx/ircg/) from Sascha Schumann.
2. SGI Static Threads Library (http://sourceforge.net/projects/state-threads/)
3. thttpd (http://www.acme.com/software/thttpd/) webserver

Installation

A detailed installation instruction can be found here (http://lxr.php.net/source/php4/ext/ircg/README.txt).

ircg_channel_mode (PHP 4 >= 4.0.5)

Set channel mode flags for user

boolean **ircg_channel_mode** (resource connection, string channel, string mode_spec, string nick) \linebreak

Set channel mode flags for *channel* on server connected to by *connection*. Mode flags are passed in *mode_spec* and are applied to the user specified by *nick*.

Mode flags are set or cleared by specifind a mode character and prepending it with a plus or minus character respectively. E.g. operator mode is granted by '+o' and revoked by '-o' passed as *mode_spec*.

ircg_disconnect (PHP 4 >= 4.0.4)

Close connection to server

boolean **ircg_disconnect** (resource connection, string reason) \linebreak

ircg_disconnect() will close a *connection* to a server previously established with ircg-pconnect().

See also: ircg_pconnect().

ircg_fetch_error_msg (PHP 4 >= 4.1.0)

Returns the error from previous ircg operation

array **ircg_fetch_error_msg** (resource connection) \linebreak

ircg_fetch_error_msg() returns the error from the last called ircg function.

> **Nota:** Errorcode is stored in first array element, errortext in second.

Esempio 1. ircg_fetch_error_msg() example

```
if (!ircg_join ($id, "#php")) {
    $error = ircg_fetch_error_msg($id);
    print ("Can't join channel #php. Errorcode:
            $error[0] Description: $error[1]");
}
```

ircg_get_username (PHP 4 >= 4.1.0)

Get username for connection

string **ircg_get_username** (int connection) \linebreak

Function **ircg_get_username()** returns the username for specified connection *connection*. Returns FALSE if *connection* died or is not valid.

ircg_html_encode (PHP 4 >= 4.0.5)

Encodes HTML preserving output

boolean **ircg_html_encode** (string html_string) \linebreak

Encodes a HTML string *html_string* for output. This feature could be usable, e.g. if someone wants to discuss about an html problem.

ircg_ignore_add (PHP 4 >= 4.0.5)

Add a user to your ignore list on a server

boolean **ircg_ignore_add** (resource connection, string nick) \linebreak

This function will add user *nick* to your ignore list on the server connected to by *connection*. By doing so you will suppress all messages from this user from being send to you.

See also: ircg_ignore_del().

ircg_ignore_del (PHP 4 >= 4.0.5)

Remove a user from your ignore list on a server

boolean **ircg_ignore_del** (resource connection, string nick) \linebreak

This function remove user *nick* from your ignore list on the server connected to by *connection*.

See also: ircg_ignore_add().

ircg_is_conn_alive (PHP 4 >= 4.0.5)

Check connection status

boolean **ircg_is_conn_alive** (resource connection) \linebreak

ircg_is_conn_alive() returns TRUE if *connection* is still alive and working or FALSE if the server no longer talks to us.

ircg_join (PHP 4 >= 4.0.4)

Join a channel on a connected server

boolean **ircg_join** (resource connection, string channel) \linebreak

Join the channel *channel* on the server connected to by *connection*.

ircg_kick (PHP 4 >= 4.0.5)

Kick a user out of a channel on server

boolean **ircg_kick** (resource connection, string channel, string nick, string reason) \linebreak

Kick user *nick* from *channel* on server connected to by *connection*. *reason* should give a short message describing why this action was performed.

ircg_lookup_format_messages (PHP 4 >= 4.0.5)

Select a set of format strings for display of IRC messages

boolean **ircg_lookup_format_messages** (string name) \linebreak

Select the set of format strings to use for display of IRC messages and events. Sets may be registered with ircg_register_format_messages(), a default set named `ircg` is always available.

See also: ircg_register_format_messages()

ircg_msg (PHP 4 >= 4.0.4)

Send message to channel or user on server

boolean **ircg_msg** (resource connection, string recipient, string message [, boolean suppress]) \linebreak

ircg_msg() will send the message to a channel or user on the server connected to by *connection*. A *recipient* starting with `#` or `&` will send the *message* to a channel, anything else will be interpreted as a username.

Setting the optional parameter *suppress* to a `TRUE` value will suppress output of your message to your own *connection*.

ircg_nick (PHP 4 >= 4.0.5)

Change nickname on server

boolean **ircg_nick** (resource connection, string nick) \linebreak

Change your nickname on the given *connection* to the one given in *nick* if possible.

Will return TRUE on success and FALSE on failure.

ircg_nickname_escape (PHP 4 >= 4.0.6)

Encode special characters in nickname to be IRC-compliant

string **ircg_nickname_escape** (string nick) \linebreak

Function **ircg_nickname_escape()** returns a decoded nickname specified by *nick* wich is IRC-compliant.

See also: ircg_nickname_unescape()

ircg_nickname_unescape (PHP 4 >= 4.0.6)

Decodes encoded nickname

string **ircg_nickname_unescape** (string nick) \linebreak

Function **ircg_nickname_unescape()** returns a decoded nickname, which is specified in *nick*.

See also: ircg_nickname_escape()

ircg_notice (PHP 4 >= 4.0.5)

Send a notice to a user on server

boolean **ircg_notice** (resource connection, string , string message) \linebreak

This function will send the *message* text to the user *nick* on the server connected to by *connection*. Query your IRC documentation of choice for the exact difference between a MSG and a NOTICE.

ircg_part (PHP 4 >= 4.0.4)

Leave a channel on server

boolean **ircg_part** (resource connection, string channel) \linebreak

Leave the channel *channel* on the server connected to by *connection*.

ircg_pconnect (PHP 4 >= 4.0.4)

Connect to an IRC server

resource **ircg_pconnect** (string username [, string server_ip [, int server_port [, string msg_format [, array ctcp_messages [, array user_settings]]]]]) \linebreak

ircg_pconnect() will try to establish a connection to an IRC server and return a connection resource handle for further use.

The only mandatory parameter is *`username`*, this will set your initial nickname on the server. *`server_ip`* and *`server_port`* are optional and default to `127.0.0.1` and `6667`.

> **Nota:** For now parameter *`server_ip`* will not do any hostname lookups and will only accept IP addresses in numerical form.
>
> Currently **ircg_pconnect()** always returns a valid handle, even if the connection failed.

You can customize the output of IRC messages and events by selection a format string set previously created with ircg_register_format_messages() by specifing the sets name in *`msg_format`*.

`ctcp_messages`

`user_settings`

See also: ircg_disconnect(), ircg_is_conn_alive(), ircg_register_format_messages().

ircg_register_format_messages (PHP 4 >= 4.0.5)

Register a set of format strings for display of IRC messages

boolean **ircg_register_format_messages** (string name, array messages) \linebreak

With **ircg_register_format_messages()** you can customize the way your IRC output looks like. You can even register different format string sets and switch between them on the fly with ircg_lookup_format_messages().

- Plain channel message
- Private message received
- Private message sent
- Some user leaves channel
- Some user enters channel
- Some user was kicked from the channel
- Topic has been changed
- Error
- Fatal error
- Join list end(?)

- Self part(?)
- Some user changes his nick
- Some user quits his connection
- Mass join begin
- Mass join element
- Mass join end
- Whois user
- Whois server
- Whois idle
- Whois channel
- Whois end
- Voice status change on user
- Operator status change on user
- Banlist
- Banlist end

- %f - from
- %t - to
- %c - channel
- %r - plain message
- %m - encoded message
- %j - js encoded message

- 1 - mod encode
- 2 - nickname decode

See also: ircg_lookup_format_messages().

ircg_set_current (PHP 4 >= 4.0.4)

Set current connection for output

boolean **ircg_set_current** (resource connection) \linebreak

Select the current connection for output in this execution context. Every output sent from the server connected to by `connection` will be copied to standard output while using default formatting or a format string set specified by ircg_register_format_messages() and selected by ircg_lookup_format_messages().

See also: ircg_register_format_messages() and ircg_lookup_format_messages().

ircg_set_file (PHP 4 >= 4.2.0)

Set logfile for connection

bool **ircg_set_file** (int connection, string path) \linebreak

Function **ircg_set_file()** specifies a logfile *path* in which all output from connection *connection* will be logged. Returns TRUE on success, otherwise FALSE.

ircg_set_on_die (PHP 4 >= 4.2.0)

Set hostaction to be execute when connection dies

bool **ircg_set_on_die** (int connection, string host, int port, string data) \linebreak

In case of the termination of connection *connection* IRCG will connect to *host* at *port* (Note: host must be an IPv4 address, IRCG does not resolve host-names due to blocking issues), send *data* to the new host connection and will wait until the remote part closes connection. This can be used to trigger a php script for example.

ircg_topic (PHP 4 >= 4.0.5)

Set topic for channel on server

boolean **ircg_topic** (resource connection, string channel, string new_topic) \linebreak

Change the topic for channel *channel* on the server connected to by *connection* to *new_topic*.

ircg_whois (PHP 4 >= 4.0.5)

Query user information for nick on server

boolean **ircg_whois** (resource connection, string nick) \linebreak

Sends a query to the connected server *connection* to send information for the specified user *nick*.

XLVIII. Java

There are two possible ways to bridge PHP and Java: you can either integrate PHP into a Java Servlet environment, which is the more stable and efficient solution, or integrate Java support into PHP. The former is provided by a SAPI module that interfaces with the Servlet server, the latter by the Java extension.

PHP 4 ext/java provides a simple and effective means for creating and invoking methods on Java objects from PHP. The JVM is created using JNI, and everything runs in-process. Build instructions for ext/java can be found in `php4/ext/java/README`.

Esempio 1. Java Example

```
<?php
  // get instance of Java class java.lang.System in PHP
  $system = new Java('java.lang.System');

  // demonstrate property access
  print 'Java version='.$system->getProperty('java.version').' <br>';
  print 'Java vendor=' .$system->getProperty('java.vendor').'  <br>';
  print 'OS='.$system->getProperty('os.name').' '.
              $system->getProperty('os.version').' on '.
              $system->getProperty('os.arch').' <br>';

  // java.util.Date example
  $formatter = new Java('java.text.SimpleDateFormat',
                        "EEEE, MMMM dd, yyyy 'at' h:mm:ss a zzzz");

  print $formatter->format(new Java('java.util.Date'));
?>
```

Esempio 2. AWT Example

```
<?php
  // This example is only intented to be run as a CGI.

  $frame  = new Java('java.awt.Frame', 'PHP');
  $button = new Java('java.awt.Button', 'Hello Java World!');

  $frame->add('North', $button);
  $frame->validate();
  $frame->pack();
  $frame->visible = True;

  $thread = new Java('java.lang.Thread');
  $thread->sleep(10000);

  $frame->dispose();
?>
```

Notes:

- `new Java()` will create an instance of a class if a suitable constructor is available. If no parameters are passed and the default constructor is useful as it provides access to classes like `java.lang.System` which expose most of their functionallity through static methods.
- Accessing a member of an instance will first look for bean properties then public fields. In other words, `print $date.time` will first attempt to be resolved as `$date.getTime()`, then as `$date.time`.
- Both static and instance members can be accessed on an object with the same syntax. Furthermore, if the java object is of type `java.lang.Class`, then static members of the class (fields and methods) can be accessed.
- Exceptions raised result in PHP warnings, and `NULL` results. The warnings may be eliminated by prefixing the method call with an "@" sign. The following APIs may be used to retrieve and reset the last error:
  - java_last_exception_get()
  - java_last_exception_clear()
- Overload resolution is in general a hard problem given the differences in types between the two languages. The PHP Java extension employs a simple, but fairly effective, metric for determining which overload is the best match.

  Additionally, method names in PHP are not case sensitive, potentially increasing the number of overloads to select from.

  Once a method is selected, the parameters are cooerced if necessary, possibly with a loss of data (example: double precision floating point numbers will be converted to boolean).
- In the tradition of PHP, arrays and hashtables may pretty much be used interchangably. Note that hashtables in PHP may only be indexed by integers or strings; and that arrays of primitive types in Java can not be sparse. Also note that these constructs are passed by value, so may be expensive in terms of memory and time.

sapi/servlet builds upon the mechanism defined by ext/java to enable the entire PHP processor to be run as a servlet. The primary advanatage of this from a PHP perspective is that web servers which support servlets typically take great care in pooling and reusing JVMs. Build instructions for the Servlet SAPI module can be found in `php4/sapi/README`. Notes:

- While this code is intended to be able to run on any servlet engine, it has only been tested on Apache's Jakarta/tomcat to date. Bug reports, success stories and/or patches required to get this code to run on other engines would be appreciated.
- PHP has a habit of changing the working directory. sapi/servlet will eventually change it back, but while PHP is running the servlet engine may not be able to load any classes from the CLASSPATH which are specified using a relative directory syntax, or find the work directory used for administration and JSP compilation tasks.

java_last_exception_clear (PHP 4 >= 4.0.2)

Clear last Java exception

void **java_last_exception_clear** (void) \linebreak

java_last_exception_get (PHP 4 >= 4.0.2)

Get last Java exception

exception **java_last_exception_get** (void) \linebreak

The following example demonstrates the usage of Java's exception handler from within PHP:

Esempio 1. Java exception handler

```
<?php
  $stack = new Java('java.util.Stack');
  $stack->push(1);

  // This should succeed
  $result = $stack->pop();
  $ex = java_last_exception_get();
  if (!$ex) print "$result\n";

  // This should fail (error suppressed by @)
  $result = @$stack->pop();
  $ex = java_last_exception_get();
  if ($ex) print $ex->toString();

  // Clear last exception
  java_last_exception_clear();
?>
```

XLIX. LDAP functions

Introduction to LDAP

LDAP is the Lightweight Directory Access Protocol, and is a protocol used to access "Directory Servers". The Directory is a special kind of database that holds information in a tree structure.

The concept is similar to your hard disk directory structure, except that in this context, the root directory is "The world" and the first level subdirectories are "countries". Lower levels of the directory structure contain entries for companies, organisations or places, while yet lower still we find directory entries for people, and perhaps equipment or documents.

To refer to a file in a subdirectory on your hard disk, you might use something like

```
/usr/local/myapp/docs
```

The forwards slash marks each division in the reference, and the sequence is read from left to right.

The equivalent to the fully qualified file reference in LDAP is the "distinguished name", referred to simply as "dn". An example dn might be.

```
cn=John Smith,ou=Accounts,o=My Company,c=US
```

The comma marks each division in the reference, and the sequence is read from right to left. You would read this dn as ..

```
country = US
organization = My Company
organizationalUnit = Accounts
commonName = John Smith
```

In the same way as there are no hard rules about how you organise the directory structure of a hard disk, a directory server manager can set up any structure that is meaningful for the purpose. However, there are some conventions that are used. The message is that you can not write code to access a directory server unless you know something about its structure, any more than you can use a database without some knowledge of what is available.

Complete code example

Retrieve information for all entries where the surname starts with "S" from a directory server, displaying an extract with name and email address.

Esempio 1. LDAP search example

```
<?php
// basic sequence with LDAP is connect, bind, search, interpret search
// result, close connection
```

```
echo "<h3>LDAP query test</h3>";
echo "Connecting ...";
$ds=ldap_connect("localhost");  // must be a valid LDAP server!
echo "connect result is ".$ds."<p>";

if ($ds) {
    echo "Binding ...";
    $r=ldap_bind($ds);     // this is an "anonymous" bind, typically
                           // read-only access
    echo "Bind result is ".$r."<p>";

    echo "Searching for (sn=S*) ...";
    // Search surname entry
    $sr=ldap_search($ds,"o=My Company, c=US", "sn=S*");
    echo "Search result is ".$sr."<p>";

    echo "Number of entires returned is ".ldap_count_entries($ds,$sr)."<p>";

    echo "Getting entries ...<p>";
    $info = ldap_get_entries($ds, $sr);
    echo "Data for ".$info["count"]." items returned:<p>";

    for ($i=0; $i<$info["count"]; $i++) {
        echo "dn is: ". $info[$i]["dn"] ."<br>";
        echo "first cn entry is: ". $info[$i]["cn"][0] ."<br>";
        echo "first email entry is: ". $info[$i]["mail"][0] ."<p>";
    }

    echo "Closing connection";
    ldap_close($ds);

} else {
    echo "<h4>Unable to connect to LDAP server</h4>";
}
?>
```

Using the PHP LDAP calls

You will need to get and compile LDAP client libraries from either the University of Michigan ldap-3.3 package or the Netscape Directory SDK 3.0. You will also need to recompile PHP with LDAP support enabled before PHP's LDAP calls will work.

Before you can use the LDAP calls you will need to know ..

- The name or address of the directory server you will use
- The "base dn" of the server (the part of the world directory that is held on this server, which could be "o=My Company,c=US")
- Whether you need a password to access the server (many servers will provide read access for an

"anonymous bind" but require a password for anything else)

The typical sequence of LDAP calls you will make in an application will follow this pattern:

```
ldap_connect()    // establish connection to server
   |
ldap_bind()       // anonymous or authenticated "login"
   |
do something like search or update the directory
and display the results
   |
ldap_close()      // "logout"
```

More Information

Lots of information about LDAP can be found at

- Netscape (http://developer.netscape.com/tech/directory/)
- University of Michigan (http://www.umich.edu/~dirsvcs/ldap/index.html)
- OpenLDAP Project (http://www.openldap.org/)
- LDAP World (http://www.innosoft.com/ldapworld)

The Netscape SDK contains a helpful Programmer's Guide in .html format.

ldap_8859_to_t61 (PHP 4 >= 4.0.2)

Translate 8859 characters to t61 characters

string **ldap_8859_to_t61** (string value) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_add (PHP 3, PHP 4 >= 4.0.0)

Add entries to LDAP directory

bool **ldap_add** (resource link_identifier, string dn, array entry) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

The **ldap_add()** function is used to add entries in the LDAP directory. The DN of the entry to be added is specified by *`dn`*. Array *`entry`* specifies the information about the entry. The values in the entries are indexed by individual attributes. In case of multiple values for an attribute, they are indexed using integers starting with 0.

entry["attribute1"] = value
entry["attribute2"][0] = value1
entry["attribute2"][1] = value2

Esempio 1. Complete example with authenticated bind

```
<?php
$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {
    // bind with appropriate dn to give update access
    $r=ldap_bind($ds,"cn=root, o=My Company, c=US", "secret");

    // prepare data
    $info["cn"]="John Jones";
    $info["sn"]="Jones";
    $info["mail"]="jonj@here.and.now";
    $info["objectclass"]="person";

    // add data to directory
    $r=ldap_add($ds, "cn=John Jones, o=My Company, c=US", $info);

    ldap_close($ds);
} else {
    echo "Unable to connect to LDAP server";
```

```
}
?>
```

ldap_bind (PHP 3, PHP 4 >= 4.0.0)

Bind to LDAP directory

bool **ldap_bind** (resource link_identifier [, string bind_rdn [, string bind_password]]) \linebreak

Binds to the LDAP directory with specified RDN and password. Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

ldap_bind() does a bind operation on the directory. *`bind_rdn`* and *`bind_password`* are optional. If not specified, anonymous bind is attempted.

ldap_close (PHP 3, PHP 4 >= 4.0.0)

Close link to LDAP server

bool **ldap_close** (resource link_identifier) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

ldap_close() closes the link to the LDAP server that's associated with the specified *`link_identifier`*.

This call is internally identical to ldap_unbind(). The LDAP API uses the call ldap_unbind(), so perhaps you should use this in preference to **ldap_close()**.

> **Nota:** This function is an alias of ldap_unbind().

ldap_compare (PHP 4 >= 4.0.2)

Compare value of attribute found in entry specified with DN

bool **ldap_compare** (resource link_identifier, string dn, string attribute, string value) \linebreak

Returns `TRUE` if *`value`* matches otherwise returns `FALSE`. Returns -1 on error.

ldap_compare() is used to compare *`value`* of *`attribute`* to value of same attribute in LDAP directory entry specified with *`dn`*.

The following example demonstrates how to check whether or not given password matches the one defined in DN specified entry.

Esempio 1. Complete example of password check

```
<?php

$ds=ldap_connect("localhost");  // assuming the LDAP server is on this host

if ($ds) {

    // bind
    if(ldap_bind($ds)) {

        // prepare data
        $dn = "cn=Matti Meikku, ou=My Unit, o=My Company, c=FI";
        $value = "secretpassword";
        $attr = "password";

        // compare value
        $r=ldap_compare($ds, $dn, $attr, $value);

        if ($r === -1) {
            echo "Error: ".ldap_error($ds);
        } elseif ($r === TRUE) {
            echo "Password correct.";
        } elseif ($r === FALSE) {
            echo "Wrong guess! Password incorrect.";
        }

    } else {
        echo "Unable to bind to LDAP server.";
    }

    ldap_close($ds);

} else {
    echo "Unable to connect to LDAP server.";
}
?>
```

Attenzione

ldap_compare() can NOT be used to compare BINARY values!

Nota: This function was added in 4.0.2.

ldap_connect (PHP 3, PHP 4 >= 4.0.0)

Connect to an LDAP server

resource **ldap_connect** ([string hostname [, int port]]) \linebreak

Returns a positive LDAP link identifier on success, or `FALSE` on error.

ldap_connect() establishes a connection to a LDAP server on a specified *`hostname`* and *`port`*. Both the arguments are optional. If no arguments are specified then the link identifier of the already opened link will be returned. If only *`hostname`* is specified, then the port defaults to 389.

If you are using OpenLDAP 2.x.x you can specify a URL instead of the hostname. To use LDAP with SSL, compile OpenLDAP 2.x.x with SSL support, configure PHP with SSL, and use ldaps://hostname/ as host parameter. The port parameter is not used when using URLs.

> **Nota:** URL and SSL support were added in 4.0.4.

ldap_count_entries (PHP 3, PHP 4 >= 4.0.0)

Count the number of entries in a search

int **ldap_count_entries** (resource link_identifier, resource result_identifier) \linebreak

Returns number of entries in the result or `FALSE` on error.

ldap_count_entries() returns the number of entries stored in the result of previous search operations. *`result_identifier`* identifies the internal ldap result.

ldap_delete (PHP 3, PHP 4 >= 4.0.0)

Delete an entry from a directory

bool **ldap_delete** (resource link_identifier, string dn) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

ldap_delete() function delete a particular entry in LDAP directory specified by *`dn`*.

ldap_dn2ufn (PHP 3, PHP 4 >= 4.0.0)

Convert DN to User Friendly Naming format

string **ldap_dn2ufn** (string dn) \linebreak

ldap_dn2ufn() function is used to turn a DN, specified by *`dn`*, into a more user-friendly form, stripping off type names.

ldap_err2str (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Convert LDAP error number into string error message

string **ldap_err2str** (int errno) \linebreak

Returns string error message.

This function returns the string error message explaining the error number *errno*. While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

See also ldap_errno() and ldap_error().

Esempio 1. Enumerating all LDAP error messages

```
<?php
  for($i=0; $i<100; $i++) {
    printf("Error $i: %s<br>\n", ldap_err2str($i));
  }
?>
```

ldap_errno (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Return the LDAP error number of the last LDAP command

int **ldap_errno** (resource link_identifier) \linebreak

Return the LDAP error number of the last LDAP command for this link.

This function returns the standardized error number returned by the last LDAP command for the given *link_identifier*. This number can be converted into a textual error message using ldap_err2str().

Unless you lower your warning level in your `php.ini` sufficiently or prefix your LDAP commands with @ (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

Esempio 1. Generating and catching an error

```
<?php
// This example contains an error, which we will catch.
$ld = ldap_connect("localhost");
$bind = ldap_bind($ld);
// syntax error in filter expression (errno 87),
// must be "objectclass=*" to work.
$res =  @ldap_search($ld, "o=Myorg, c=DE", "objectclass");
if (!$res) {
    printf("LDAP-Errno: %s<br>\n", ldap_errno($ld));
    printf("LDAP-Error: %s<br>\n", ldap_error($ld));
```

```
    die("Argh!<br>\n");
}
$info = ldap_get_entries($ld, $res);
printf("%d matching entries.<br>\n", $info["count"]);
?>
```

See also ldap_err2str() and ldap_error().

ldap_error (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Return the LDAP error message of the last LDAP command

string **ldap_error** (resource link_identifier) \linebreak

Returns string error message.

This function returns the string error message explaining the error generated by the last LDAP command for the given *link_identifier* While LDAP errno numbers are standardized, different libraries return different or even localized textual error messages. Never check for a specific error message text, but always use an error number to check.

Unless you lower your warning level in your `php.ini` sufficiently or prefix your LDAP commands with `@` (at) characters to suppress warning output, the errors generated will also show up in your HTML output.

See also ldap_err2str() and ldap_errno().

ldap_explode_dn (PHP 3, PHP 4 >= 4.0.0)

Splits DN into its component parts

array **ldap_explode_dn** (string dn, int with_attrib) \linebreak

ldap_explode_dn() function is used to split the DN returned by ldap_get_dn() and breaks it up into its component parts. Each part is known as Relative Distinguished Name, or RDN. **ldap_explode_dn()** returns an array of all those components. *with_attrib* is used to request if the RDNs are returned with only values or their attributes as well. To get RDNs with the attributes (i.e. in attribute=value format) set *with_attrib* to 0 and to get only values set it to 1.

ldap_first_attribute (PHP 3, PHP 4 >= 4.0.0)

Return first attribute

string **ldap_first_attribute** (resource link_identifier, resource result_entry_identifier, int ber_identifier) \linebreak

Returns the first attribute in the entry on success and `FALSE` on error.

Similar to reading entries, attributes are also read one by one from a particular entry. **ldap_first_attribute()** returns the first attribute in the entry pointed by the *result_entry_identifier*. Remaining attributes are retrieved by calling ldap_next_attribute() successively. *ber_identifier* is the identifier to internal memory location pointer. It is passed by reference. The same *ber_identifier* is passed to the ldap_next_attribute() function, which modifies that pointer.

See also ldap_get_attributes()

ldap_first_entry (PHP 3, PHP 4 >= 4.0.0)

Return first result id

resource **ldap_first_entry** (resource link_identifier, resource result_identifier) \linebreak

Returns the result entry identifier for the first entry on success and FALSE on error.

Entries in the LDAP result are read sequentially using the **ldap_first_entry()** and ldap_next_entry() functions. **ldap_first_entry()** returns the entry identifier for first entry in the result. This entry identifier is then supplied to **lap_next_entry()** routine to get successive entries from the result.

See also ldap_get_entries().

ldap_first_reference (PHP 4 >= 4.0.5)

Return first reference

resource **ldap_first_reference** (resource link, resource result) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_free_result (PHP 3, PHP 4 >= 4.0.0)

Free result memory

bool **ldap_free_result** (resource result_identifier) \linebreak

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

ldap_free_result() frees up the memory allocated internally to store the result and pointed by the *result_identifier*. All result memory will be automatically freed when the script terminates.

Typically all the memory allocated for the ldap result gets freed at the end of the script. In case the script is making successive searches which return large result sets, **ldap_free_result()** could be called to keep the runtime memory usage by the script low.

ldap_get_attributes (PHP 3, PHP 4 >= 4.0.0)

Get attributes from a search result entry

array **ldap_get_attributes** (resource link_identifier, resource result_entry_identifier) \linebreak

Returns a complete entry information in a multi-dimensional array on success and `FALSE` on error.

ldap_get_attributes() function is used to simplify reading the attributes and values from an entry in the search result. The return value is a multi-dimensional array of attributes and values.

Having located a specific entry in the directory, you can find out what information is held for that entry by using this call. You would use this call for an application which "browses" directory entries and/or where you do not know the structure of the directory entries. In many applications you will be searching for a specific attribute such as an email address or a surname, and won't care what other data is held.

return_value["count"] = number of attributes in the entry
return_value[0] = first attribute
return_value[n] = nth attribute

return_value["attribute"]["count"] = number of values for attribute
return_value["attribute"][0] = first value of the attribute
return_value["attribute"][i] = (i+1)th value of the attribute

Esempio 1. Show the list of attributes held for a particular directory entry

```
// $ds is the link identifier for the directory

// $sr is a valid search result from a prior call to
// one of the ldap directory search calls

$entry = ldap_first_entry($ds, $sr);

$attrs = ldap_get_attributes($ds, $entry);

echo $attrs["count"]." attributes held for this entry:<p>";

for ($i=0; $i<$attrs["count"]; $i++)
    echo $attrs[$i]."<br>";
```

See also ldap_first_attribute() and ldap_next_attribute()

ldap_get_dn (PHP 3, PHP 4 >= 4.0.0)

Get the DN of a result entry

string **ldap_get_dn** (resource link_identifier, resource result_entry_identifier) \linebreak

Returns the DN of the result entry and `FALSE` on error.

ldap_get_dn() function is used to find out the DN of an entry in the result.

ldap_get_entries (PHP 3, PHP 4 >= 4.0.0)

Get all result entries

array **ldap_get_entries** (resource link_identifier, resource result_identifier) \linebreak

Returns a complete result information in a multi-dimenasional array on success and `FALSE` on error.

ldap_get_entries() function is used to simplify reading multiple entries from the result, specified with *`result_identifier`*, and then reading the attributes and multiple values. The entire information is returned by one function call in a multi-dimensional array. The structure of the array is as follows.

The attribute index is converted to lowercase. (Attributes are case-insensitive for directory servers, but not when used as array indices.)

```
return_value["count"] = number of entries in the result
return_value[0] : refers to the details of first entry

return_value[i]["dn"] =  DN of the ith entry in the result

return_value[i]["count"] = number of attributes in ith entry
return_value[i][j] = jth attribute in the ith entry in the result

return_value[i]["attribute"]["count"] = number of values for
                                        attribute in ith entry
return_value[i]["attribute"][j] = jth value of attribute in ith entry
```

See also ldap_first_entry() and ldap_next_entry()

ldap_get_option (PHP 4 >= 4.0.4)

Get the current value for given option

bool **ldap_get_option** (resource link_identifier, int option, mixed retval) \linebreak

Sets *`retval`* to the value of the specified option. Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN. These are described in draft-ietf-ldapext-ldap-c-api-xx.txt (http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt)

> **Nota:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4

Esempio 1. Check protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_get_option($ds, LDAP_OPT_PROTOCOL_VERSION, $version))
    echo "Using protocol version $version";
else
    echo "Unable to determine protocol version";
```

See also ldap_set_option().

ldap_get_values (PHP 3, PHP 4 >= 4.0.0)

Get all values from a result entry

array **ldap_get_values** (resource link_identifier, resource result_entry_identifier, string attribute) \linebreak

Returns an array of values for the attribute on success and FALSE on error.

ldap_get_values() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *result_entry_identifier*. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

This call needs a *result_entry_identifier*, so needs to be preceded by one of the ldap search calls and one of the calls to get an individual entry.

You application will either be hard coded to look for certain attributes (such as "surname" or "mail") or you will have to use the ldap_get_attributes() call to work out what attributes exist for a given entry.

LDAP allows more than one entry for an attribute, so it can, for example, store a number of email addresses for one person's directory entry all labeled with the attribute "mail"

return_value["count"] = number of values for attribute
return_value[0] = first value of attribute
return_value[i] = ith value of attribute

Esempio 1. List all values of the "mail" attribute for a directory entry

```
// $ds is a valid link identifier for a directory server

// $sr is a valid search result from a prior call to
//     one of the ldap directory search calls

// $entry is a valid entry identifier from a prior call to
//        one of the calls that returns a directory entry

$values = ldap_get_values($ds, $entry,"mail");

echo $values["count"]." email addresses for this entry.<p>";

for ($i=0; $i < $values["count"]; $i++)
    echo $values[$i]."<br>";
```

ldap_get_values_len (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Get all binary values from a result entry

array **ldap_get_values_len** (resource link_identifier, resource result_entry_identifier, string attribute) \linebreak

Returns an array of values for the attribute on success and `FALSE` on error.

ldap_get_values_len() function is used to read all the values of the attribute in the entry in the result. entry is specified by the *`result_entry_identifier`*. The number of values can be found by indexing "count" in the resultant array. Individual values are accessed by integer index in the array. The first index is 0.

This function is used exactly like ldap_get_values() except that it handles binary data and not string data.

Nota: This function was added in 4.0.

ldap_list (PHP 3, PHP 4 >= 4.0.0)

Single-level search

resource **ldap_list** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]]) \linebreak

Returns a search result identifier or `FALSE` on error.

ldap_list() performs the search for a specified *`filter`* on the directory with the scope LDAP_SCOPE_ONELEVEL.

LDAP_SCOPE_ONELEVEL means that the search should only return information that is at the level immediately below the *base_dn* given in the call. (Equivalent to typing "ls" and getting a list of files and folders in the current working directory.)

This call takes 5 optional parameters. See ldap_search() notes.

> **Nota:** These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

Esempio 1. Produce a list of all organizational units of an organization

```
// $ds is a valid link identifier for a directory server

$basedn = "o=My Company, c=US";
$justthese = array("ou");

$sr=ldap_list($ds, $basedn, "ou=*", $justthese);

$info = ldap_get_entries($ds, $sr);

for ($i=0; $i<$info["count"]; $i++)
    echo $info[$i]["ou"][0] ;
```

> **Nota:** From 4.0.5 on it's also possible to do parallel searches. See ldap_search() for details.

ldap_mod_add (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Add attribute values to current attributes

bool **ldap_mod_add** (resource link_identifier, string dn, array entry) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

This function adds attribute(s) to the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level additions are done by the ldap_add() function.

ldap_mod_del (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Delete attribute values from current attributes

bool **ldap_mod_del** (resource link_identifier, string dn, array entry) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

This function removes attribute(s) from the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level deletions are done by the ldap_delete() function.

ldap_mod_replace (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Replace attribute values with new ones

bool **ldap_mod_replace** (resoure link_identifier, string dn, array entry) \linebreak

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

This function replaces attribute(s) from the specified *dn*. It performs the modification at the attribute level as opposed to the object level. Object-level modifications are done by the ldap_modify() function.

ldap_modify (PHP 3, PHP 4 >= 4.0.0)

Modify an LDAP entry

bool **ldap_modify** (resource link_identifier, string dn, array entry) \linebreak

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

ldap_modify() function is used to modify the existing entries in the LDAP directory. The structure of the entry is same as in ldap_add().

ldap_next_attribute (PHP 3, PHP 4 >= 4.0.0)

Get the next attribute in result

string **ldap_next_attribute** (resource link_identifier, resource result_entry_identifier, resource ber_identifier) \linebreak

Returns the next attribute in an entry on success and FALSE on error.

ldap_next_attribute() is called to retrieve the attributes in an entry. The internal state of the pointer is maintained by the *ber_identifier*. It is passed by reference to the function. The first call to **ldap_next_attribute()** is made with the *result_entry_identifier* returned from ldap_first_attribute().

See also ldap_get_attributes()

ldap_next_entry (PHP 3, PHP 4 >= 4.0.0)

Get next result entry

resource **ldap_next_entry** (resource link_identifier, resource result_entry_identifier) \linebreak

Returns entry identifier for the next entry in the result whose entries are being read starting with ldap_first_entry(). If there are no more entries in the result then it returns `FALSE`.

ldap_next_entry() function is used to retrieve the entries stored in the result. Successive calls to the **ldap_next_entry()** return entries one by one till there are no more entries. The first call to **ldap_next_entry()** is made after the call to ldap_first_entry() with the *`result_entry_identifier`* as returned from the ldap_first_entry().

See also ldap_get_entries()

ldap_next_reference (PHP 4 >= 4.0.5)

Get next reference

resource **ldap_next_reference** (resource link, resource entry) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_parse_reference (PHP 4 >= 4.0.5)

Extract information from reference entry

bool **ldap_parse_reference** (resource link, resource entry, array referrals) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_parse_result (PHP 4 >= 4.0.5)

Extract information from result

bool **ldap_parse_result** (resource link, resource result, int errcode, string matcheddn, string errmsg, array referrals) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_read (PHP 3, PHP 4 >= 4.0.0)

Read an entry

resource **ldap_read** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]]) \linebreak

Returns a search result identifier or `FALSE` on error.

ldap_read() performs the search for a specified *`filter`* on the directory with the scope LDAP_SCOPE_BASE. So it is equivalent to reading an entry from the directory.

An empty filter is not allowed. If you want to retrieve absolutely all information for this entry, use a filter of "objectClass=*". If you know which entry types are used on the directory server, you might use an appropriate filter such as "objectClass=inetOrgPerson".

This call takes 5 optional parameters. See ldap_search() notes.

> **Nota:** These optional parameters were added in 4.0.2: *`attrsonly`*, *`sizelimit`*, *`timelimit`*, *`deref`*.

From 4.0.5 on it's also possible to do parallel searches. See ldap_search() for details.

ldap_rename (PHP 4 >= 4.0.5)

Modify the name of an entry

bool **ldap_rename** (resource link_identifier, string dn, string newrdn, string newparent, bool deleteoldrdn) \linebreak

The entry specified by *`dn`* is renamed/moved. The new RDN is specified by *`newrdn`* and the new parent/superior entry is specified by *`newparent`*. If the parameter *`deleteoldrdn`* is `TRUE` the old RDN value(s) is removed, else the old RDN value(s) is retained as non-distinguished values of the entry. Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

> **Nota:** This function currently only works with LDAPv3. You may have to use ldap_set_option() prior to binding to use LDAPv3. This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.5.

ldap_search (PHP 3, PHP 4 >= 4.0.0)

Search LDAP tree

resource **ldap_search** (resource link_identifier, string base_dn, string filter [, array attributes [, int attrsonly [, int sizelimit [, int timelimit [, int deref]]]]]) \linebreak

Returns a search result identifier or `FALSE` on error.

ldap_search() performs the search for a specified filter on the directory with the scope of LDAP_SCOPE_SUBTREE. This is equivalent to searching the entire directory. *base_dn* specifies the base DN for the directory.

There is a optional fourth parameter, that can be added to restrict the attributes and values returned by the server to just those required. This is much more efficient than the default action (which is to return all attributes and their associated values). The use of the fourth parameter should therefore be considered good practice.

The fourth parameter is a standard PHP string array of the required attributes, eg array("mail","sn","cn") Note that the "dn" is always returned irrespective of which attributes types are requested.

Note too that some directory server hosts will be configured to return no more than a preset number of entries. If this occurs, the server will indicate that it has only returned a partial results set. This occurs also if the sixth parameter *sizelimit* has been used to limit the count of fetched entries.

The fifth parameter *attrsonly* should be set to 1 if only attribute types are wanted. If set to 0 both attributes types and attribute values are fetched which is the default behaviour.

With the sixth parameter *sizelimit* it is possible to limit the count of entries fetched. Setting this to 0 means no limit. NOTE: This parameter can NOT override server-side preset sizelimit. You can set it lower though.

The seventh parameter *timelimit* sets the number of seconds how long is spend on the search. Setting this to 0 means no limit. NOTE: This parameter can NOT override server-side preset timelimit. You can set it lower though.

The eigth parameter *deref* specifies how aliases should be handled during the search. It can be one of the following:

- LDAP_DEREF_NEVER - (default) aliases are never dereferenced.
- LDAP_DEREF_SEARCHING - aliases should be dereferenced during the search but not when locating the base object of the search.
- LDAP_DEREF_FINDING - aliases should be dereferenced when locating the base object but not during the search.
- LDAP_DEREF_ALWAYS - aliases should be dereferenced always.

> **Nota:** These optional parameters were added in 4.0.2: *attrsonly*, *sizelimit*, *timelimit*, *deref*.

The search filter can be simple or advanced, using boolean operators in the format described in the LDAP doumentation (see the Netscape Directory SDK

(http://developer.netscape.com/docs/manuals/directory/41/ag/find.htm) for full information on filters).

The example below retrieves the organizational unit, surname, given name and email address for all people in "My Company" where the surname or given name contains the substring $person. This example uses a boolean filter to tell the server to look for information in more than one attribute.

Esempio 1. LDAP search

```
// $ds is a valid link identifier for a directory server

// $person is all or part of a person’s name, eg "Jo"

$dn = "o=My Company, c=US";
$filter="(|(sn=$person*)(givenname=$person*))";
$justthese = array( "ou", "sn", "givenname", "mail");

$sr=ldap_search($ds, $dn, $filter, $justthese);

$info = ldap_get_entries($ds, $sr);

print $info["count"]." entries returned<p>";
```

From 4.0.5 on it’s also possible to do parallel searches. To do this you use an array of link identifiers, rather than a single identifier, as the first argument. If you don’t want the same base DN and the same filter for all the searches, you can also use an array of base DNs and/or an array of filters. Those arrays must be of the same size as the link identifier array since the first entries of the arrays are used for one search, the second entries are used for another, and so on. When doing parallel searches an array of search result identifiers is returned, except in case of error, then the entry corresponding to the search will be `FALSE`. This is very much like the value normally returned, except that a result identifier is always returned when a search was made. There are some rare cases where the normal search returns `FALSE` while the parallel search returns an identifier.

ldap_set_option (PHP 4 >= 4.0.4)

Set the value of the given option

bool **ldap_set_option** (resource link_identifier, int option, mixed newval) \linebreak

Sets the value of the specified option to be *newval*. Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento. on error.

The parameter *option* can be one of: LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION, LDAP_OPT_ERROR_NUMBER, LDAP_OPT_REFERRALS, LDAP_OPT_RESTART, LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING, LDAP_OPT_MATCHED_DN, LDAP_OPT_SERVER_CONTROLS, LDAP_OPT_CLIENT_CONTROLS. Here’s a brief description, see draft-ietf-ldapext-ldap-c-api-xx.txt

(http://www.openldap.org/devel/cvsweb.cgi/~checkout~/doc/drafts/draft-ietf-ldapext-ldap-c-api-xx.txt) for
details.

The options LDAP_OPT_DEREF, LDAP_OPT_SIZELIMIT, LDAP_OPT_TIMELIMIT, LDAP_OPT_PROTOCOL_VERSION and LDAP_OPT_ERROR_NUMBER have integer value, LDAP_OPT_REFERRALS and LDAP_OPT_RESTART have boolean value, and the options LDAP_OPT_HOST_NAME, LDAP_OPT_ERROR_STRING and LDAP_OPT_MATCHED_DN have string value. The first example illustrates their use. The options LDAP_OPT_SERVER_CONTROLS and LDAP_OPT_CLIENT_CONTROLS require a list of controls, this means that the value must be an array of controls. A control consists of an *oid* identifying the control, an optional *value*, and an optional flag for *criticality*. In PHP a control is given by an array containing an element with the key *oid* and string value, and two optional elements. The optional elements are key *value* with string value and key *iscritical* with boolean value. *iscritical* defaults to `FALSE` if not supplied. See also the second example below.

> **Nota:** This function is only available when using OpenLDAP 2.x.x OR Netscape Directory SDK x.x, and was added in PHP 4.0.4.

Esempio 1. Set protocol version

```
// $ds is a valid link identifier for a directory server
if (ldap_set_option($ds, LDAP_OPT_PROTOCOL_VERSION, 3))
    echo "Using LDAPv3";
else
    echo "Failed to set protocol version to 3";
```

Esempio 2. Set server controls

```
// $ds is a valid link identifier for a directory server
// control with no value
$ctrl1 = array("oid" => "1.2.752.58.10.1", "iscritical" => TRUE);
// iscritical defaults to FALSE
$ctrl2 = array("oid" => "1.2.752.58.1.10", "value" => "magic");
// try to set both controls
if (!ldap_set_option($ds, LDAP_OPT_SERVER_CONTROLS, array($ctrl1, $ctrl2)))
    echo "Failed to set server controls";
```

See also ldap_get_option().

ldap_set_rebind_proc (PHP 4 >= 4.2.0)

Set a callback function to do re-binds on referral chasing.

bool **ldap_set_rebind_proc** (resource link, string callback) \linebreak

> **Attenzione**
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_sort (PHP 4 >= 4.2.0)

Sort LDAP result entries

bool **ldap_sort** (resource link, resource result, string sortfilter) \linebreak

> **Attenzione**
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_start_tls (PHP 4 >= 4.2.0)

Start TLS

bool **ldap_start_tls** (resource link) \linebreak

> **Attenzione**
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_t61_to_8859 (PHP 4 >= 4.0.2)

Translate t61 characters to 8859 characters

string **ldap_t61_to_8859** (string value) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ldap_unbind (PHP 3, PHP 4 >= 4.0.0)

Unbind from LDAP directory

bool **ldap_unbind** (resource link_identifier) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

ldap_unbind() function unbinds from the LDAP directory.

L. Funzioni di Mail

La funzione mail() consente di inviare mail.

Direttive di configurazione per Mail

`SMTP` string

Nome DNS o indirizzo IP del server SMTP che PHP sotto Windows deve usare per spedire posta elettronica con la funzione mail().

`sendmail_from` string

Quale campo "From:" devono avere i messaggi inviati da PHP sotto Windows.

`sendmail_path` string

Dove trovare il programma **sendmail**, solitamente `/usr/sbin/sendmail` oppure `/usr/lib/sendmail`. **configure** cerca di trovare il file e lo imposta di default, ma se non riesce a localizzarlo, lo si può impostare qui.

I sistemi che non usano sendmail devono impostare questa direttiva al wrapper che i rispettivi sistemi di posta offrono, se esistenti. Per esempio, gli utenti di Qmail (http://www.qmail.org/) possono normalmente impostarla a `/var/qmail/bin/sendmail`.

ezmlm_hash (PHP 3>= 3.0.17, PHP 4 >= 4.0.2)

Calcola il valore hash che occorre a EZMLM

int **ezmlm_hash** (string addr) \linebreak

ezmlm_hash() calcola il valore hash che occorre quando si mantengono mailing list EZMLM in un database MySQL.

Esempio 1. Calcolare l'hash e iscrivere un utente

```
$utente = "pippo@example.com";
$hash = ezmlm_hash ($utente);
$query = sprintf ("INSERT INTO esempio VALUES (%s, '%s')", $hash, $utente);
$db->query($query); // tramite l'interfaccia db PHPLIB
```

mail (PHP 3, PHP 4 >= 4.0.0)

Invio mail

bool **mail** (string a, string oggetto, string messaggio [, string header_addizionali [, string parametri_addizionali]]) \linebreak

mail() invia automaticamente il messaggio specificato in `messaggio` al destinatario specificato in `a`. Destinatari multipli possono essere specificati mettendo una virgola tra ogni indirizzo in `a`. Email con allegati e tipi speciali di contenuto possono essere spedite usando questa funzione. Questo è possibile tramite la codifica MIME. Per maggiori informazioni, fare riferimento a un articolo Zend (http://www.zend.com/zend/spotlight/sendmimeemailpart1.php) o alle Classi Mime del PEAR (http://pear.php.net/get/Mail_Mime).

Le seguenti RFC possono essere di aiuto: RFC 1896 (http://www.ietf.org/rfc/rfc1896.txt), RFC 2045 (http://www.ietf.org/rfc/rfc2045.txt), RFC 2046 (http://www.ietf.org/rfc/rfc2046.txt), RFC 2047 (http://www.ietf.org/rfc/rfc2047.txt), RFC 2048 (http://www.ietf.org/rfc/rfc2048.txt) e RFC 2049 (http://www.ietf.org/rfc/rfc2049.txt).

mail() restituisce `TRUE` se la mail è stata spedita con successo, altrimenti restituisce `FALSE`.

Esempio 1. Inviare mail.

```
mail("pippo@example.com", "Oggetto", "Linea 1\nLinea 2\nLinea 3");
```

Se viene passata come parametro una quarta stringa, questa stringa viene inserita alla fine dell'intestazione (header). Ciò viene tipicamente usato per aggiungere intestazioni supplementari.

Intestazioni multiple supplementari sono separate da un carattere di "a capo" (sia newline che carriage return).

> **Nota:** È necessario usare `\r\n` per separare le intestazioni, alcuni mail transfer agent sotto Unix potrebbero funzionare anche solo con un singolo newline (`\n`). L'intestazione Cc: è case sensitive e deve essere scritta esattamente così `Cc:` sui sistemi Win32. L'intestazione Bcc: non è supportata dai sistemi Win32.

Esempio 2. Invio di mail con intestazioni supplementari.

```
mail("nessuno@example.com", "oggetto", $messaggio,
     "From: webmaster@$SERVER_NAME\r\n"
    ."Reply-To: webmaster@$SERVER_NAME\\r\n"
    ."X-Mailer: PHP/" . phpversion());
```

Con il parametro *`parametri_addizionali`* è possibile impostare parametri addizionali a linea di comando per il programma configurato per inviare mail usando `sendmail_path`. Per esempio si può impostare il corretto valore per envelope sender di sendmail. Potrebbe essere necessario aggiungere l'utente che ha in esecuzione il server web alla configurazione di sendmail per prevenire l'aggiunta dell'intestazione 'X-Warning' quando si imposta envelope sender in questo modo.

Esempio 3. Invio di mail con intestazioni supplementari e impostazione dei parametri addizionali a linea di comando.

```
mail("nessuno@example.com", "oggetto", $messaggio,
     "From: webmaster@$SERVER_NAME", "-fwebmaster@$SERVER_NAME");
```

> **Nota:** Questo quinto parametro è stato aggiunto in PHP 4.0.5.

È possibile costruire messaggi complessi utilizzando la tecnica di concatenazione delle stringhe.

Esempio 4. Invio di mail complessa.

```
/* destinatari */
$destinatari .= "Maria <maria@example.com>" . ", " ; // notare la virgola
$destinatari .= "Enrica <enrica@example.com>";

/* oggetto */
$oggetto = "Promemoria compleanni per Agosto";
```

```
/* messaggio */
$messaggio = '
<html>
<head>
 <title>Promemoria compleanni per Agosto</title>
</head>
<body>
<p>Questi sono i compleanni di Agosto!</p>
<table>
 <tr>
  <th>Persona</th><th>Giorno</th><th>Mese</th><th>Anno</th>
 </tr>
 <tr>
  <td>Giovanni</td><td>3</td><td>Agosto</td><td>1970</td>
  <td>Sara</td><td>17</td><td>Agosto</td><td>1973</td>
 </tr>
</table>
</body>
</html>
';

/* Per inviare email in formato HTML, si deve impostare l'intestazione Content-
type. */
$intestazioni  = "MIME-Version: 1.0\r\n";
$intestazioni .= "Content-type: text/html; charset=iso-8859-1\r\n";

/* intestazioni addizionali */
$intestazioni .= "From: Promemoria Compleanni <compleanni@example.com>\r\n";

$intestazioni .= "Cc: archiviocompleanni@example.com\r\n";
$intestazioni .= "Bcc: controllocompleanni@example.com\r\n";

/* ed infine l'invio */
mail($destinatari, $oggetto, $messaggio, $intestazioni);
```

Nota: Assicurarsi di non avere nessun carattere di newline nei parametri *a* o `oggetto`, o la mail non verrà spedita correttamente.

LI. mailparse functions

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

mailparse_determine_best_xfer_encoding (4.1.0 - 4.2.0 only)

Figures out the best way of encoding the content read from the file pointer fp, which must be seek-able

int **mailparse_determine_best_xfer_encoding** (resource fp) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_create (4.1.0 - 4.2.0 only)

Returns a handle that can be used to parse a message

int **mailparse_msg_create** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_extract_part (4.1.0 - 4.2.0 only)

Extracts/decodes a message section. If callbackfunc is not specified, the contents will be sent to "stdout"

void **mailparse_msg_extract_part** (resource rfc2045, string msgbody [, string callbackfunc]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_extract_part_file (4.1.0 - 4.2.0 only)

Extracts/decodes a message section, decoding the transfer encoding

string **mailparse_msg_extract_part_file** (resource rfc2045, string filename [, string callbackfunc]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_free (4.1.0 - 4.2.0 only)

Frees a handle allocated by mailparse_msg_crea

void **mailparse_msg_free** (resource rfc2045buf) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_get_part (4.1.0 - 4.2.0 only)

Returns a handle on a given section in a mimemessage

int **mailparse_msg_get_part** (resource rfc2045, string mimesection) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_get_part_data (4.1.0 - 4.2.0 only)

Returns an associative array of info about the message

array **mailparse_msg_get_part_data** (resource rfc2045) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_get_structure (4.1.0 - 4.2.0 only)

Returns an array of mime section names in the supplied message

array **mailparse_msg_get_structure** (resource rfc2045) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_parse (4.1.0 - 4.2.0 only)

Incrementally parse data into buffer

void **mailparse_msg_parse** (resource rfc2045buf, string data) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_msg_parse_file (4.1.0 - 4.2.0 only)

Parse file and return a resource representing the structure

resource **mailparse_msg_parse_file** (string filename) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_rfc822_parse_addresses (4.1.0 - 4.2.0 only)

Parse addresses and returns a hash containing that data

array **mailparse_rfc822_parse_addresses** (string addresses) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_stream_encode (4.1.0 - 4.2.0 only)

Streams data from source file pointer, apply encoding and write to destfp

bool **mailparse_stream_encode** (resource sourcefp, resource destfp, string encoding) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mailparse_uudecode_all (PHP 4 4.2.0 only)

Scans the data from fp and extract each embedded uuencoded file. Returns an array listing filename information

array **mailparse_uudecode_all** (resource fp) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

LII. Funzioni Matematiche

Introduzione

Queste funzioni matematiche operano esclusivamente nel range dei tipi di dato integer e float del computer. (questo corrisponde attualmente ai tipi di dati long e double del C) Se si ha necessità di lavorare con numeri più grandi, fare riferimento alle funzioni matematiche a precisione arbitraria.

Costanti Matematiche

I seguenti valori sono definiti come costanti nel PHP dall'estensione matematica del linguaggio:

Tabella 1. Costanti Matematiche

Costante	Valore	Descrizione
M_PI	3.14159265358979323846	Pi
M_E	2.7182818284590452354	e
M_LOG2E	1.4426950408889634074	log_2 e
M_LOG10E	0.43429448190325182765	log_10 e
M_LN2	0.69314718055994530942	log_e 2
M_LN10	2.30258509299404568402	log_e 10
M_PI_2	1.57079632679489661923	pi/2
M_PI_4	0.78539816339744830962	pi/4
M_1_PI	0.31830988618379067154	1/pi
M_2_PI	0.63661977236758134308	2/pi
M_SQRTPI	1.77245385090551602729	sqrt(pi) [4.0.2]
M_2_SQRTPI	1.12837916709551257390	2/sqrt(pi)
M_SQRT2	1.41421356237309504880	sqrt(2)
M_SQRT3	1.73205080756887729352	sqrt(3) [4.0.2]
M_SQRT1_2	0.70710678118654752440	1/sqrt(2)
M_LNPI	1.14472988584940017414	log_e(pi) [4.0.2]
M_EULER	0.57721566490153286061	Costante di Eulero [4.0.2]

Soltanto M_PI è disponibile nelle versioni precedenti alla PHP 4.0.0 (compresa). Tutte le rimanenti costanti sono disponibili a partire dal PHP 4.0.0. Le costanti indicate con [4.0.2] sono state aggiunte nel PHP 4.0.2.

abs (PHP 3, PHP 4 >= 4.0.0)

Valore assoluto

mixed **abs** (mixed numero) \linebreak

Restituisce il valore assoluto di un numero. Se l'argomento della funzione è di tipo float, il valore restituito è float, altrimenti restituisce un integer (perché float di solito ha un range di valori più grande di integer).

Esempio 1. Esempio di abs()

```
$abs = abs(-4.2); // $abs = 4.2; (double/float)
$abs2 = abs(5);   // $abs2 = 5; (integer)
$abs3 = abs(-5);  // $abs3 = 5; (integer)
```

acos (PHP 3, PHP 4 >= 4.0.0)

Arco coseno

float **acos** (float arg) \linebreak

Restituisce l'arco coseno di `arg` in radianti.

Vedere anche `arg`, asin() e atan().

acosh (PHP 4 >= 4.1.0)

Inverso del coseno iperbolico

float **acosh** (float arg) \linebreak

Restituisce l'inverso del coseno iperbolico di `arg`, cioè il valore il cui coseno iperbolico vale `arg`.

> **Nota:** Questa funzione non è implementata su piattaforme Windows

Vedere anche acos(), asin() e atan().

asin (PHP 3, PHP 4 >= 4.0.0)

Arco seno

float **asin** (float arg) \linebreak

Restituisce l'arco seno di arg in radianti.

Vedere anche asinh(), acos() e atan().

asinh (PHP 4 >= 4.1.0)

Inverso del seno iperbolico

float **asinh** (float arg) \linebreak

Restituisce l'inverso del seno iperbolico di `arg`, cioè il valore il cui seno iperbolico vale `arg`

> **Nota:** Questa funzione non è implementata su piattaforme Windows

vedere anche asin(), acos() e atan().

atan (PHP 3, PHP 4 >= 4.0.0)

Arco tangente

float **atan** (float arg) \linebreak

Restituisce l'arco tangente di arg in radianti.

Vedere anche atanh(), asin() e acos().

atan2 (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

arco tangente di due variabili

float **atan2** (float y, float x) \linebreak

Questa funzione calcola l'arco tangente delle due variabili `x` e `y`. È simile al calcolo dell'arco tangente di `y` / `x`, eccetto per il fatto che viene tenuto in consideazione il segno di entrambi gli argomenti per determinare il quadrante del risultato.

La funzione restituisce il risultato in radianti, compreso fra -PI e PI (inclusi).

Vedere anche acos() e atan().

atanh (PHP 4 >= 4.1.0)

Inverso della tangente iperbolica

float **atanh** (float arg) \linebreak

Restituisce l'inverso della tangente iperbolica di *`arg`*, cioè il valore la cui tangente iperbolica vale *`arg`*.

> **Nota:** Questa funzione non è implementata su piattaforme Windows

See also atan(), asin() e acos().

base_convert (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Converte un numero fra basi arbitrarie

string **base_convert** (string numero, int base_di_partenza, int base_di_arrivo) \linebreak

Restituisce una stringa contenente *`numero`* rappresentata in base *`base_di_arrivo`*. La base in cui *`numero`* è dato è specificata da *`base_di_partenza`*. Entrambe *`base_di_partenza`* e *`base_di_arrivo`* devono essere comprese fra 2 e 36, inclusi. Cifre in numeri con una base maggiore di 10 saranno rappresentati con le lettere a-z, con a significante 10, b significante 11 e z significante 35.

Esempio 1. base_convert()

```
$binario = base_convert ($esadecimale, 16, 2);
```

bindec (PHP 3, PHP 4 >= 4.0.0)

Da binario a decimale

int **bindec** (string stringa_binaria) \linebreak

Restituisce il decimale equivalente al numero binario rappresentato dall'argomento stringa_binaria.

bindec() converte un binario in integer. Il più grande numero che può essere convertito è 31 volte la cifra 1 oppure 2147483647 espresso in formato decimale.

Vedere anche la funzione decbin().

ceil (PHP 3, PHP 4 >= 4.0.0)

arroronda le frazioni all'intero superiore

float **ceil** (float numero) \linebreak

Restituisce il primo intero più grande di *`numero`*, se necessario. Il valore restituito da **ceil()** è ancora di tipo float, poiché la gamma di valori del tipo float è solitamente più grande di quella del tipo int.

Esempio 1. Esempio di ceil()

```
$cinque = ceil(4.3);   // $cinque = 5.0;
$dieci = ceil(9.999); // $dieci = 10.0;
```

Vedere anche floor() e round().

cos (PHP 3, PHP 4 >= 4.0.0)

Coseno

float **cos** (float arg) \linebreak

Restituisce il coseno di arg in radianti.

Vedere anche sin() e tan().

cosh (PHP 4 >= 4.1.0)

Coseno iperbolico

float **cosh** (float arg) \linebreak

Restituisce il coseno iperbolico di `arg`, definito come `(exp(arg) + exp(-arg))/2`.

Vedere anche cos(), acosh(), sin() and tan().

decbin (PHP 3, PHP 4 >= 4.0.0)

Da decimale a binario

string **decbin** (int numero) \linebreak

Restituisce una stringa contenente una rappresentazione binaria di un dato argomento `numero`. Il più grande numero che può essere convertito è 4294967295 in decimale, risultante in una stringa composta da 32 volte la cifra 1.

Vedere anche la funzione bindec().

dechex (PHP 3, PHP 4 >= 4.0.0)

Da decimale a esadecimale

string **dechex** (int numero) \linebreak

Restituisce una stringa contenente una rappresentazione esadecimale di un dato argomento `numero`. Il più grande numero che puù essere convertito è 2147483647 in decimale risultante in "7fffffff".

Vedere anche la funzione hexdec().

decoct (PHP 3, PHP 4 >= 4.0.0)

Da decimale a ottale

string **decoct** (int numero) \linebreak

Restituisce una stringa contenente una rappresentazione in ottale di un dato argomento `numero`. Il più grande numero che può essere convertito è 2147483647 in decimale risultante in "17777777777". Vedere anche octdec().

Vedere anche octdec().

deg2rad (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Converte il numero dato in gradi nell'equivalente espresso in radianti

double **deg2rad** (double numero) \linebreak

Questa funzione converte `numero` da gradi al valore equivalente espresso in radianti.

Vedere anche rad2deg().

exp (PHP 3, PHP 4 >= 4.0.0)

e elevato alla potenza di ...

float **exp** (float arg) \linebreak

Restituisce e elevato alla potenza di `arg`.

Vedere anche pow().

expm1 (PHP 4 >= 4.1.0)

Restituisce exp(numero) - 1, computato in maniera tale da essere accurato anche se il valore del numero è vicino a zero

float **expm1** (float number) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

floor (PHP 3, PHP 4 >= 4.0.0)

Arrotonda le frazioni all'intero inferiore

float **floor** (float numero) \linebreak

Restituisce il primo intero più piccolo di `numero`, se necessario. Il valore restituito da **floor()** è ancora di tipo float, poiché la gamma di valori del tipo float è solitamente più grande di quella del tipo int.

Esempio 1. Esempio di floor()

```
$quattro = floor(4.3); // $quattro = 4.0;
$nove = floor(9.999);  // $nove = 9.0;
```

Vedere anche ceil() e round().

getrandmax (PHP 3, PHP 4 >= 4.0.0)

Mostra il più grande numero casuale disponibile

int **getrandmax** (void) \linebreak

Restituisce il valore massimo che può essere restituito da una chiamata alla funzione rand().

Vedere anche rand(), srand() mt_rand(), mt_srand() e mt_getrandmax().

hexdec (PHP 3, PHP 4 >= 4.0.0)

Da esadecimale a decimale

int **hexdec** (string stringa_esadecimale) \linebreak

Restituisce l'equivalente decimale di un numero esadecimale rappresentato dall'argomento stringa_esadecimale. **hexdec()** converte una stringa esadecimale in un numero decimale. Il più grande numero che può essere convertito è 7fffffff o 2147483647 espresso in decimale.

hexdec() sostituisce ogni carattere non esadecimale che incontra con `0`. In questo modo, tutti gli zeri a sinistra sono ignorati, ma gli zeri a destra sono valutati.

Esempio 1. esempio di hexdec()

```
var_dump(hexdec("See"));
var_dump(hexdec("ee"));
// entrambi stampano "int(238)"

var_dump(hexdec("that"));
var_dump(hexdec("a0"));
// entrambi stampano int(160)
```

Vedere anche dechex().

hypot (PHP 4 >= 4.1.0)

Restituisce sqrt(num1*num1 + num2*num2)

float **hypot** (float num1, float num2) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

is_finite (PHP 4 >= 4.2.0)

bool **is_finite** (float val) \linebreak

Restituisce `TRUE` se *`val`* è un numero finito valido all'interno del range dei float, come definito da PHP su questa piattaforma.

is_infinite (PHP 4 >= 4.2.0)

bool **is_infinite** (float val) \linebreak

Restituisce `TRUE` se *`val`* è infinito (positivo o negativo), come il risultato di `log(0)` o ogni altro valore troppo grande per essere contenuto nel range dei float di una determinata piattaforma.

is_nan (PHP 4 >= 4.2.0)

bool **is_nan** (float val) \linebreak

Restituisce `TRUE` se *`val`* 'non è un numero', come il risultato di `acos(1.01)`.

lcg_value (PHP 4 >= 4.0.0)

Generatore combinato lineare congruenziale

float **lcg_value** (void) \linebreak

lcg_value() restituisce uno pseudo numero casuale nell'intervallo (0, 1). La funzione combina due GC con periodo di 2^31 - 85 e 2^31 - 249. Il periodo di questa funzione è uguale al prodotto dei due primi.

log (PHP 3, PHP 4 >= 4.0.0)

Logaritmo naturale

float **log** (float arg) \linebreak

Restituisce il logaritmo naturale di arg.

log10 (PHP 3, PHP 4 >= 4.0.0)

Logaritmo base-10

float **log10** (float arg) \linebreak

Restituisce il logaritmo in base-10 di *arg*.

log1p (PHP 4 >= 4.1.0)

Restituisce log(1 + numero), computato in maniera tale da essere accurato anche se il valore del numero è vicino a zero

float **log1p** (float number) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

max (PHP 3, PHP 4 >= 4.0.0)

Trova il valore massimo

mixed **max** (mixed arg1, mixed arg2, mixed argn) \linebreak

max() restituisce il numericamente più grande dei valori dati come parametro.

Se il primo parametro è un array, **max()** restituisce il massimo dei valori di tale array. Se il primo parametro è un integer, string o double, si ha bisogno almeno di due parametri e **max()** restituisce il maggiore di tali valori. Si può confrontare un numero illimitato di valori.

Se uno o più valori sono float, tutti i valori saranno considerati come float, e verrà restituito un float. Se nessuno dei valori è double, tutti verranno considerati come integer, e verrà restituito un integer.

min (PHP 3, PHP 4 >= 4.0.0)

Trova il valore minimo

mixed **min** (mixed arg1, mixed arg2, mixed argn) \linebreak number **min** (array numeri) \linebreak

min() restituisce il numericamente più piccolo dei valori dati come parametro.

Nella prima variante, si ha bisogno di almeno due parametri e **min()** restituisce il minimo fra i valori. Si può confrontare un numero illimitato di valori.

Nella seconda variante, **min()** restituisce il più piccolo valore in *numeri*.

Se uno o più valori sono float, tutti i valori saranno considerati come float, e verrà restituito un float. Se nessuno dei valori è float, tutti verranno considerati come integer, e verrà restituito un integer.

mt_getrandmax (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Mostra il più grande valore casuale disponibile

int **mt_getrandmax** (void) \linebreak

Restituisce il massimo valore che può essere restituito da una chiamata alla funzione mt_rand().

Vedere anche mt_rand(), mt_srand(), rand(), srand() e getrandmax().

mt_rand (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Genera un valore casuale migliore

int **mt_rand** ([int min, int max]) \linebreak

Molti generatori di numeri casuali di vecchie libc hanno caratteristiche dubbie o sconosciute e sono lenti. Di default, PHP usa il generatore di numeri casuali libc con la funzione rand(). La funzione **mt_rand()** è un sostituto per questa. Usa un generatore di numeri casuali con caratteristiche conosciute, il Mersenne Twister, che assicura numeri casuali che dovrebbero essere adatti per scopi crittografici e (vedere la homepage per i dettagli) e che è mediamente quattro volte più veloce di libc. L'Homepage del Mersenne Twister può essere trovata qui: http://www.math.keio.ac.jp/~matumoto/emt.html, e una versione ottimizzata del codice sorgente di MT è disponibile a questo inidrizzo: http://www.scp.syr.edu/~marc/hawk/twister.html (http://www.scp.syr.edu/~marc/hawk/twister.html).

Se invocata senza i parametri opzionali *min*, *max*, **mt_rand()** restituisce un valore pseudo-casuale compreso fra 0 e `RAND_MAX`. Se ad esempio si desidera un numero casuale compreso fra 5 e 15 (inclusi), usare `mt_rand (5, 15)`.

Ricordarsi di inizializzare il generatore di numeri casuali usando mt_srand().

> **Nota:** Nelle versioni precedenti la 3.0.7 il significato di *max* era *range*. Per ottenere lo stesso risultato in queste vecchie versioni un breve esempio dovrebbe essere il seguente: `mt_rand (5, 11)`, si otterrà un numero casuale compreso fra 5 e 15.

Vedere anche mt_srand(), mt_getrandmax(), srand(), rand() e getrandmax().

mt_srand (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Inizializza un generatore di numeri casuali migliore

void **mt_srand** (int seme) \linebreak

Inizializza il generatore di numeri casuali con il parametro *seme*.

```
// inizializza usando i microsecondi
function crea_seme() {
    list($usec, $sec) = explode(' ', microtime());
    return (float) $sec + ((float) $usec * 100000);
}
mt_srand(crea_seme());
$valorecasuale = mt_rand();
```

Vedere anche mt_rand(), mt_getrandmax(), srand(), rand() e getrandmax().

number_format (PHP 3, PHP 4 >= 4.0.0)

Formatta un numero raggruppando le migliaia

string **number_format** (float numero [, int decimali [, string separatore_decimali [, string separatore_migliaia]]]) \linebreak

number_format() restituisce una versione formattata di *numero*. Questa funzione accetta uno, due o quattro parametri (non tre):

Se viene dato un solo parametro, *numero* sarà formattato senza decimali, ma con una virgola (",") fra ogni gruppo di migliaia.

Se vengono dati due parametri, *numero* sarà formattato con *decimali* decimali con un punto (".") davanti, e una virgola (",") fra ciascun gruppo di migliaia.

Se vengono dati tutti e quattro i parametri, *numero* sarà formattato con *decimali* decimali, *separatore_decimali* invece di un punto (".") prima dei decimali e *separatore_migliaia* invece di una virgola (",") fra ciascun gruppo di migliaia.

> **Nota:** Viene usato solo il primo carattere di *separatore_migliaia*. Per esempio, se si usa `foo` come *separatore_migliaia* sul numero `1000`, **number_format()** restituirà `1f000`.

Esempio 1. Esempio di number_format()

Si noti che la notazione Francese solitamente usa due decimali, la virgola (',') come separatore decimale, e lo spazio (' ') come separatore delle migliaia. Questo si può ottenere usando questa linea:

```
<?php
```

```
    $numero = 1234.56;

    // notazione inglese (predefinita)
    $numero_in_formato_inglese = number_format($numero);
    // 1,234.56

    // notazione francese
    $numero_in_formato_francese = number_format($numero, 2, ’,’, ’ ’);
    // 1 234,56

    $numero = 1234.5678;

    // notazione inglese senza separatore delle migliaia
    $numero_in_formato_inglese = number_format($numero, 2, ’.’, ”);
    // 1234.56

?>
```

Nota: Vedere anche: sprintf(), printf() e sscanf().

octdec (PHP 3, PHP 4 >= 4.0.0)

Da ottale a decimale

int **octdec** (string stringa_ottale) \linebreak

Restituisce l’equivalente decimale del numero ottale rappresentato dall’argomento stringa_ottale. OctDec converte una stringa ottale in un numero decimale. Il più grande numero che può essere convertito è 17777777777 o 2147483647 in decimale.

Vedere anche decoct().

pi (PHP 3, PHP 4 >= 4.0.0)

Restituisce il valore di pi

double **pi** (void) \linebreak

Restituisce una appossimazione di pi. Il valore restituito è un float e ha precisione secondo la direttiva precision del file `php.ini`, che ha come valore predefinito `14`. Inoltre, si può usare la costante `M_PI` che permette di ottenere risultati identici all’uso di **pi()**.

```
echo pi(); // 3.1415926535898
```

```
echo M_PI; // 3.1415926535898
```

pow (PHP 3, PHP 4 >= 4.0.0)

Espressione esponenziale

float **pow** (float base, float esp) \linebreak

Restituisce `base` elevato alla potenza di `esp`. Se possibile, questa funzione restituisce un integer.

Se la potenza non può essere computata, viene generato un errore, e **pow()** restituirà `FALSE`.

Esempio 1. Alcuni esempi di pow()

```
<?php

var_dump( pow(2,8) ); // int(256)
echo pow(-1,20); // 1
echo pow(0, 0); // 1

echo pow(-1, 5.5); // errore

?>
```

Attenzione

Nel PHP 4.0.6 e precedenti, **pow()** restituiva sempre un float e non generava alcun errore.

Vedere anche exp() e sqrt().

rad2deg (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Converte un numero in radianti nell'equivalente numero in gradi

double **rad2deg** (double numero) \linebreak

Questa funzione converte `numero` da radianti a gradi.

Vedere anche deg2rad().

rand (PHP 3, PHP 4 >= 4.0.0)

Genera un valore casuale

int **rand** ([int min, int max]) \linebreak

Se chiamata senza i parametri opzionali *`min`*, *`max`*, **rand()** restituisce un valore pseudo casuale compreso fra 0 e `RAND_MAX`. Se ad esempio si desidera un numero casuale compreso fra 5 e 15 (inclusi) usare `rand (5, 15)`.

Ricordarsi di inizializzare il generatore di numeri casuali usando srand().

> **Nota:** Nelle versioni precedenti la 3.0.7 il significato di *`max`* era *`range`*. Per ottenere lo stesso risultato in queste vecchie versioni un breve esempio dovrebbe essere il seguente: `rand (5, 11)`, si otterrà un numero casuale compreso fra 5 e 15.

Vedere anche srand(), getrandmax(), mt_rand(), mt_srand() e mt_getrandmax().

round (PHP 3, PHP 4 >= 4.0.0)

Arrotonda un numero non intero

double **round** (double val [, int precisione]) \linebreak

Restituisce il valore arrotondato di *`val`* con la *`precisione`* specificata (numero di cifre dopo il punto decimale). *`precision`* può anche essere negativa o zero (predefinito).

> **Cautela**
>
> Il PHP non maneggia correttamente stringhe quali `"12,300.2"` in maniera predefinita. Fare riferimento a conversione da stringhe.

```
$foo = round (3.4);  // $foo == 3.0
$foo = round (3.5);  // $foo == 4.0
$foo = round (3.6);  // $foo == 4.0
$foo = round(3.6, 0);  // equivalente alla riga sopra

$foo = round (1.95583, 2);  // $foo == 1.96

$foo = round(1241757, -3);  // $foo == 1242000
```

> **Nota:** Il parametro *`precisione`* è disponibile solo nel PHP 4.

Vedere anche ceil() e floor().

sin (PHP 3, PHP 4 >= 4.0.0)

Seno

float **sin** (float arg) \linebreak

Restituisce il seno di arg in radianti.

Vedere anche cos() e tan().

sinh (PHP 4 >= 4.1.0)

Seno iperbolico

float **sinh** (float arg) \linebreak

Restituisce il seno iperbolico di *arg*, definito come `(exp(arg) - exp(-arg))/2`.

Vedere anche sin(), asinh(), cos() e tan().

sqrt (PHP 3, PHP 4 >= 4.0.0)

Radice quadrata

float **sqrt** (float arg) \linebreak

Restituisce la radice quadrata di *arg*.

Vedere anche pow().

srand (PHP 3, PHP 4 >= 4.0.0)

inizializza il generatore di numeri casuali

void **srand** (int seme) \linebreak

Inizializza il generatore di num casuali con *seme*.

```
// inizializza usando i microsecondi
function crea_seme() {
    list($usec, $sec) = explode(’ ’, microtime());
    return (float) $sec + ((float) $usec * 100000);
}
srand((double)microtime()*1000000);
$valorecasuale = rand();
```

Vedere anche rand(), getrandmax(), mt_rand(), mt_srand() e mt_getrandmax().

tan (PHP 3, PHP 4 >= 4.0.0)

Tangente

float **tan** (float arg) \linebreak

Restituisce la tangente di arg in radianti.

Vedere anche sin() e cos().

tanh (PHP 4 >= 4.1.0)

Tangente iperbolica

float **tanh** (float arg) \linebreak

Restituisce la tangente iperbolica di *`arg`*, definita come `sinh(arg)/cosh(arg)`.

Vedere anche tan(), atanh(), sin() e cos().

LIII. Multi-Byte String Functions

Introduction

There are many languages in which all characters can be expressed by single byte. Multi-byte character codes are used to express many characters for many languages. `mbstring` is developed to handle Japanese characters. However, many `mbstring` functions are able to handle character encoding other than Japanese.

A multi-byte character encoding represents single character with consecutive bytes. Some character encoding has shift(escape) sequences to start/end multi-byte character strings. Therefore, a multi-byte character string may be destroyed when it is divided and/or counted unless multi-byte character encoding safe method is used. This module provides multi-byte character safe string functions and other utility functions such as conversion functions.

Since PHP is basically designed for ISO-8859-1, some multi-byte character encoding does not work well with PHP. Therefore, it is important to set `mbstring.internal_encoding` to a character encoding that works with PHP.

PHP4 Character Encoding Requirements

- Per byte encoding
- Single byte characters in range of `00h-7fh` which is compatible with `ASCII`
- Multi-byte characters without `00h-7fh`

These are examples of internal character encoding that works with PHP and does NOT work with PHP.

```
Character encodings work with PHP:
ISO-8859-*, EUC-JP, UTF-8

Character encodings do NOT work with PHP:
JIS, SJIS
```

Character encoding, that does not work with PHP, may be converted with `mbstring`'s HTTP input/output conversion feature/function.

> **Nota:** SJIS should not be used for internal encoding unless the reader is familiar with parser/compiler, character encoding and character encoding issues.

> **Nota:** If you use database with PHP, it is recommended that you use the same character encoding for both database and `internal encoding` for ease of use and better performance.
>
> If you are using PostgreSQL, it supports character encoding that is different from backend

character encoding. See the PostgreSQL manual for details.

How to Enable mbstring

mbstring is an extended module. You must enable module with configure script. Refer to the Install section for details.

The following configure options are related to mbstring module.

- --enable-mbstring : Enable mbstring functions. This option is required to use mbstring functions.
- --enable-mbstr-enc-trans : Enable HTTP input character encoding conversion using mbstring conversion engine. If this feature is enabled, HTTP input character encoding may be converted to mbstring.internal_encoding automatically.

HTTP Input and Output

HTTP input/output character encoding conversion may convert binary data also. Users are supposed to control character encoding conversion if binary data is used for HTTP input/output.

If enctype for HTML form is set to multipart/form-data, mbstring does not convert character encoding in POST data. If it is the case, strings are needed to be converted to internal character encoding.

- HTTP Input

  There is no way to control HTTP input character conversion from PHP script. To disable HTTP input character conversion, it has to be done in php.ini.

  Esempio 1. Disable HTTP input conversion in php.ini

  ```
  ;; Disable HTTP Input conversion
  mbstring.http_input = pass
  ```

  When using PHP as an Apache module, it is possible to override PHP ini setting per Virtual Host in httpd.conf or per directory with .htaccess. Refer to the Configuration section and Apache Manual for details.

- HTTP Output

  There are several ways to enable output character encoding conversion. One is using php.ini, another is using ob_start() with mb_output_handler() as ob_start callback function.

  Nota: For PHP3-i18n users, mbstring's output conversion differs from PHP3-i18n. Character

encoding is converted using output buffer.

Esempio 2. php.ini setting example

```
;; Enable output character encoding conversion for all PHP pages

;; Enable Output Buffering
output_buffering    = On

;; Set mb_output_handler to enable output conversion
output_handler      = mb_output_handler
```

Esempio 3. Script example

```
<?php

// Enable output character encoding conversion only for this page

// Set HTTP output character encoding to SJIS
mb_http_output('SJIS');

// Start buffering and specify "mb_output_handler" as
// callback function
ob_start('mb_output_handler');

?>
```

Supported Character Encoding

Currently, the following character encoding is supported by `mbstring` module. Caracter encoding may be specified for `mbstring` functions' `encoding` parameter.

The following character encoding is supported in this PHP extension :

`UCS-4`, `UCS-4BE`, `UCS-4LE`, `UCS-2`, `UCS-2BE`, `UCS-2LE`, `UTF-32`, `UTF-32BE`, `UTF-32LE`, `UCS-2LE`, `UTF-16`, `UTF-16BE`, `UTF-16LE`, `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`, `SJIS`, `eucJP-win`, `SJIS-win`, `ISO-2022-JP`, `JIS`, `ISO-8859-1`, `ISO-8859-2`, `ISO-8859-3`, `ISO-8859-4`, `ISO-8859-5`, `ISO-8859-6`, `ISO-8859-7`, `ISO-8859-8`, `ISO-8859-9`, `ISO-8859-10`,

ISO-8859-13, ISO-8859-14, ISO-8859-15, byte2be, byte2le, byte4be, byte4le, BASE64, 7bit, 8bit and UTF7-IMAP.

php.ini entry, which accepts encoding name, accepts "auto" and "pass" also. mbstring functions, which accepts encoding name, and accepts "auto".

If "pass" is set, no character encoding conversion is performed.

If "auto" is set, it is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS".

See also mb_detect_order()

> **Nota:** "Supported character encoding" does not mean that it works as internal character code.

php.ini settings

- mbstring.internal_encoding defines default internal character encoding.
- mbstring.http_input defines default HTTP input character encoding.
- mbstring.http_output defines default HTTP output character encoding.
- mbstring.detect_order defines default character code detection order. See also mb_detect_order().
- mbstring.substitute_character defines character to substitute for invalid character encoding.

Web Browsers are supposed to use the same character encoding when submitting form. However, browsers may not use the same character encoding. See mb_http_input() to detect character encoding used by browsers.

If enctype is set to multipart/form-data in HTML forms, mbstring does not convert character encoding in POST data. The user must convert them in the script, if conversion is needed.

Although, browsers are smart enough to detect character encoding in HTML. charset is better to be set in HTTP header. Change default_charset according to character encoding.

Esempio 4. php.ini setting example

```
;; Set default internal encoding
;; Note: Make sure to use character encoding works with PHP
mbstring.internal_encoding    = UTF-8  ; Set internal encoding to UTF-8

;; Set default HTTP input character encoding
;; Note: Script cannot change http_input setting.
mbstring.http_input           = pass    ; No conversion.
mbstring.http_input           = auto    ; Set HTTP input to auto
                                ; "auto" is expanded to "ASCII,JIS,UTF-8,EUC-
JP,SJIS"
mbstring.http_input           = SJIS    ; Set HTTP2 input to  SJIS
mbstring.http_input           = UTF-8,SJIS,EUC-JP ; Specify order
```

```
;; Set default HTTP output character encoding
mbstring.http_output          = pass    ; No conversion
mbstring.http_output          = UTF-8   ; Set HTTP output encoding to UTF-8

;; Set default character encoding detection order
mbstring.detect_order         = auto    ; Set detect order to auto
mbstring.detect_order         = ASCII,JIS,UTF-8,SJIS,EUC-JP ; Specify order

;; Set default substitute character
mbstring.substitute_character = 12307   ; Specify Unicode value
mbstring.substitute_character = none    ; Do not print character
mbstring.substitute_character = long    ; Long Example: U+3000,JIS+7E7E
```

Esempio 5. php.ini setting for EUC-JP users

```
;; Disable Output Buffering
output_buffering      = Off

;; Set HTTP header charset
default_charset       = EUC-JP

;; Set HTTP input encoding conversion to auto
mbstring.http_input   = auto

;; Convert HTTP output to EUC-JP
mbstring.http_output  = EUC-JP

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Esempio 6. php.ini setting for SJIS users

```
;; Enable Output Buffering
output_buffering     = On

;; Set mb_output_handler to enable output conversion
output_handler       = mb_output_handler

;; Set HTTP header charset
default_charset      = Shift_JIS

;; Set http input encoding conversion to auto
mbstring.http_input  = auto
```

```
;; Convert to SJIS
mbstring.http_output = SJIS

;; Set internal encoding to EUC-JP
mbstring.internal_encoding = EUC-JP

;; Do not print invalid characters
mbstring.substitute_character = none
```

Overload of PHP string functions by mbstring functions with multibyte support

Because almost PHP application written for language using single-byte character encoding, there are some difficulties for multibyte string handling including japanese. Almost PHP string functions such as substr() do not support multibyte string.

Multibyte extension (mbstring) has some PHP string functions with multibyte support (ex. substr() supports mb_substr()).

Multibyte extension (mbstring) also supports 'function overloading' to add multibyte string functionality without code modification. Using function overloading, some PHP string functions will be oveloaded multibyte string functions. For example, mb_substr() is called instead of substr() if function overloading is enabled. Function overload makes easy to port application supporting only single-byte encoding for multibyte application.

`mbstring.func_overload` in `php.ini` should be set some positive value to use function overloading. The value should specify the category of overloading functions, sbould be set 1 to enable mail function overloading. 2 to enable string functions, 4 to regular expression functions. For example, if is set for 7, mail, strings, regex functions should be overloaded. The list of overloaded functions are shown in below.

Tabella 1. Functions to be overloaded

value of mbstring.func_overload	original function	overloaded function
1	mail()	mb_send_mail()
2	strlen()	mb_strlen()
2	strpos()	mb_strpos()
2	strrpos()	mb_strrpos()
2	substr()	mb_substr()
4	ereg()	mb_ereg()
4	eregi()	mb_eregi()
4	ereg_replace()	mb_ereg_replace()
4	eregi_replace()	mb_eregi_replace()
4	split()	mb_split()

Basics for Japanese multi-byte character

Most Japanese characters need more than 1 byte per character. In addition, several character encoding schemas are used under a Japanese environment. There are EUC-JP, Shift_JIS(SJIS) and ISO-2022-JP(JIS) character encoding. As Unicode becomes popular, UTF-8 is used also. To develop Web applications for a Japanese environment, it is important to use the character set for the task in hand, whether HTTP input/output, RDBMS and E-mail.

- Storage for a character can be up to six bytes
- A multi-byte character is usually twice of the width compared to single-byte characters. Wider characters are called "zen-kaku" - meaning full width, narrower characters are called "han-kaku" - meaning half width. "zen-kaku" characters are usually fixed width.
- Some character encoding defines shift(escape) sequence for entering/exiting multi-byte character strings.
- ISO-2022-JP must be used for SMTP/NNTP.
- "i-mode" web site is supposed to use SJIS.

References

Multi-byte character encoding and its related issues are very complex. It is impossible to cover in sufficient detail here. Please refer to the following URLs and other resources for further readings.

- Unicode/UTF/UCS/etc

  `http://www.unicode.org/`

- Japanese/Korean/Chinese character information

  `ftp://ftp.ora.com/pub/examples/nutshell/ujip/doc/cjk.inf`

mb_convert_encoding (PHP 4 >= 4.0.6)

Convert character encoding

string **mb_convert_encoding** (string str, string to-encoding [, mixed from-encoding]) \linebreak

mb_convert_encoding() converts character encoding of string *str* from *from-encoding* to *to-encoding*.

str : String to be converted.

from-encoding is specified by character code name before conversion. it can be array or string - comma separated enumerated list. If it is not specified, the internal encoding will be used.

Esempio 1. mb_convert_encoding() example

```
/* Convert internal character encoding to SJIS */
$str = mb_convert_encoding($str, "SJIS");

/* Convert EUC-JP to UTF-7 */
$str = mb_convert_encoding($str, "UTF-7", "EUC-JP");

/* Auto detect encoding from JIS, eucjp-win, sjis-win, then convert str to UCS-
2LE */
$str = mb_convert_encoding($str, "UCS-2LE", "JIS, eucjp-win, sjis-win");

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
$str = mb_convert_encoding($str, "EUC-JP", "auto");
```

See also: mb_detect_order().

mb_convert_kana (PHP 4 >= 4.0.6)

Convert "kana" one from another ("zen-kaku" ,"han-kaku" and more)

string **mb_convert_kana** (string str, string option [, mixed encoding]) \linebreak

mb_convert_kana() performs "han-kaku" - "zen-kaku" conversion for string *str*. It returns converted string. This function is only useful for Japanese.

option is conversion option. Default value is "KV".

encoding is character encoding. If it is omitted, internal character encoding is used.

```
Applicable Conversion Options

option : Specify with conversion of following options. Default "KV"
"r" :  Convert "zen-kaku" alphabets to "han-kaku"
"R" :  Convert "han-kaku" alphabets to "zen-kaku"
```

```
       "n" :  Convert "zen-kaku" numbers to "han-kaku"
       "N" :  Convert "han-kaku" numbers to "zen-kaku"
       "a" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
       "A" :  Convert "zen-kaku" alphabets and numbers to "han-kaku"
       (Characters included in "a", "A" options are
       U+0021 - U+007E excluding U+0022, U+0027, U+005C, U+007E)
       "s" :  Convert "zen-kaku" space to "han-kaku" (U+3000 -> U+0020)
       "S" :  Convert "han-kaku" space to "zen-kaku" (U+0020 -> U+3000)
       "k" :  Convert "zen-kaku kata-kana" to "han-kaku kata-kana"
       "K" :  Convert "han-kaku kata-kana" to "zen-kaku kata-kana"
       "h" :  Convert "zen-kaku hira-gana" to "han-kaku kata-kana"
       "H" :  Convert "han-kaku kata-kana" to "zen-kaku hira-gana"
       "c" :  Convert "zen-kaku kata-kana" to "zen-kaku hira-gana"
       "C" :  Convert "zen-kaku hira-gana" to "zen-kaku kata-kana"
       "V" :  Collapse voiced sound notation and convert them into a charac-
ter. Use with "K","H"
```

Esempio 1. mb_convert_kana() example

```
/* Convert all "kana" to "zen-kaku" "kata-kana" */
$str = mb_convert_kana($str, "KVC");

/* Convert "han-kaku" "kata-kana" to "zen-kaku" "kata-kana"
   and "zen-kaku" alpha-numeric to "han-kaku" */
$str = mb_convert_kana($str, "KVa");
```

mb_convert_variables (PHP 4 >= 4.0.6)

Convert character code in variable(s)

string **mb_convert_variables** (string to-encoding, mixed from-encoding, mixed vars) \linebreak

mb_convert_variables() convert character encoding of variables *`vars`* in encoding *`from-encoding`* to encoding *`to-encoding`*. It returns character encoding before conversion for success, `FALSE` for failure.

mb_convert_variables() join strings in Array or Object to detect encoding, since encoding detection tends to fail for short strings. Therefore, it is impossible to mix encoding in single array or object.

It *`from-encoding`* is specified by array or comma separated string, it tries to detect encoding from *`from-coding`*. When *`encoding`* is omitted, `detect_order` is used.

`vars (3rd and larger)` is reference to variable to be converted. String, Array and Object are accepted. **mb_convert_variables()** assumes all parameters have the same encoding.

Esempio 1. mb_convert_variables() example

```
/* Convert variables $post1, $post2 to internal encoding */
$interenc = mb_internal_encoding();
$inputenc = mb_convert_variables($interenc, "ASCII,UTF-8,SJIS-win", $post1, $post2);
```

mb_decode_mimeheader (PHP 4 >= 4.0.6)

Decode string in MIME header field

string **mb_decode_mimeheader** (string str) \linebreak

mb_decode_mimeheader() decodes encoded-word string *str* in MIME header.

It returns decoded string in internal character encoding.

See also mb_encode_mimeheader().

mb_decode_numericentity (PHP 4 >= 4.0.6)

Decode HTML numeric string reference to character

string **mb_decode_numericentity** (string str, array convmap [, string encoding]) \linebreak

Convert numeric string reference of string *str* in specified block to character. It returns converted string.

array is array to specifies code area to convert.

encoding is character encoding. If it is omitted, internal character encoding is used.

Esempio 1. *convmap* example

```
$convmap = array (
   int start_code1, int end_code1, int offset1, int mask1,
   int start_code2, int end_code2, int offset2, int mask2,
   ........
   int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN,
// then convert value to numeric string reference.
```

See also: mb_encode_numericentity().

mb_detect_encoding (PHP 4 >= 4.0.6)

Detect character encoding

string **mb_detect_encoding** (string str [, mixed encoding-list]) \linebreak

mb_detect_encoding() detects character encoding in string *str*. It returns detected character encoding.

encoding-list is list of character encoding. Encoding order may be specified by array or comma separated list string.

If *encoding_list* is omitted, detect_order is used.

Esempio 1. mb_detect_encoding() example

```
/* Detect character encoding with current detect_order */
echo mb_detect_encoding($str);

/* "auto" is expanded to "ASCII,JIS,UTF-8,EUC-JP,SJIS" */
echo mb_detect_encoding($str, "auto");

/* Specify encoding_list character encoding by comma separated list */
echo mb_detect_encoding($str, "JIS, eucjp-win, sjis-win");

/* Use array to specify encoding_list  */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
echo mb_detect_encoding($str, $ary);
```

See also: mb_detect_order().

mb_detect_order (PHP 4 >= 4.0.6)

Set/Get character encoding detection order

array **mb_detect_order** ([mixed encoding-list]) \linebreak

mb_detect_order() sets automatic character encoding detection order to *encoding-list*. It returns `TRUE` for success, `FALSE` for failure.

encoding-list is array or comma separated list of character encoding. ("auto" is expanded to "ASCII, JIS, UTF-8, EUC-JP, SJIS")

If *encoding-list* is omitted, it returns current character encoding detection order as array.

This setting affects mb_detect_encoding() and mb_send_mail().

> **Nota:** `mbstring` currently implements following encoding detection filters. If there is a invalid byte sequence for following encoding, encoding detection will fail.

Nota: `UTF-8`, `UTF-7`, `ASCII`, `EUC-JP`,`SJIS`, `eucJP-win`, `SJIS-win`, `JIS`, `ISO-2022-JP`

For `ISO-8859-*`, `mbstring` always detects as `ISO-8859-*`.

For `UTF-16`, `UTF-32`, `UCS2` and `UCS4`, encoding detection will fail always.

Esempio 1. Useless detect order example

```
; Always detect as ISO-8859-1
detect_order = ISO-8859-1, UTF-8

; Always detect as UTF-8, since ASCII/UTF-7 values are
; valid for UTF-8
detect_order = UTF-8, ASCII, UTF-7
```

Esempio 2. mb_detect_order() examples

```
/* Set detection order by enumerated list */
mb_detect_order("eucjp-win,sjis-win,UTF-8");

/* Set detection order by array */
$ary[] = "ASCII";
$ary[] = "JIS";
$ary[] = "EUC-JP";
mb_detect_order($ary);

/* Display current detection order */
echo implode(", ", mb_detect_order());
```

See also mb_internal_encoding(), mb_http_input(), mb_http_output() mb_send_mail().

mb_encode_mimeheader (PHP 4 >= 4.0.6)

Encode string for MIME header

string **mb_encode_mimeheader** (string str [, string charset [, string transfer-encoding [, string linefeed]]]) \linebreak

mb_encode_mimeheader() converts string *str* to encoded-word for header field. It returns converted string in ASCII encoding.

charset is character encoding name. Default is `ISO-2022-JP`.

transfer-encoding is transfer encoding. It should be one of "B" (Base64) or "Q" (Quoted-Printable). Default is "B".

linefeed is end of line marker. Default is "\r\n" (CRLF).

Esempio 1. mb_convert_kana() example

```
$name = ""; // kanji
$mbox = "kru";
$doma = "gtinn.mon";
$addr = mb_encode_mimeheader($name, "UTF-7", "Q") . " <" . $mbox . "@" . $doma . ">";
echo $addr;
```

See also mb_decode_mimeheader().

mb_encode_numericentity (PHP 4 >= 4.0.6)

Encode character to HTML numeric string reference

string **mb_encode_numericentity** (string str, array convmap [, string encoding]) \linebreak

mb_encode_numericentity() converts specified character codes in string *str* from HTML numeric character reference to character code. It returns converted string.

array is array specifies code area to convert.

encoding is character encoding.

Esempio 1. *convmap* example

```
$convmap = array (
 int start_code1, int end_code1, int offset1, int mask1,
 int start_code2, int end_code2, int offset2, int mask2,
 ........
 int start_codeN, int end_codeN, int offsetN, int maskN );
// Specify Unicode value for start_codeN and end_codeN
// Add offsetN to value and take bit-wise 'AND' with maskN, then
// it converts value to numeric string reference.
```

Esempio 2. mb_encode_numericentity() example

```
/* Convert Left side of ISO-8859-1 to HTML numeric character reference */
$convmap = array(0x80, 0xff, 0, 0xff);
```

```
$str = mb_encode_numericentity($str, $convmap, "ISO-8859-1");

/* Convert user defined SJIS-win code in block 95-104 to numeric
   string reference */
$convmap = array(
       0xe000, 0xe03e, 0x1040, 0xffff,
       0xe03f, 0xe0bb, 0x1041, 0xffff,
       0xe0bc, 0xe0fa, 0x1084, 0xffff,
       0xe0fb, 0xe177, 0x1085, 0xffff,
       0xe178, 0xe1b6, 0x10c8, 0xffff,
       0xe1b7, 0xe233, 0x10c9, 0xffff,
       0xe234, 0xe272, 0x110c, 0xffff,
       0xe273, 0xe2ef, 0x110d, 0xffff,
       0xe2f0, 0xe32e, 0x1150, 0xffff,
       0xe32f, 0xe3ab, 0x1151, 0xffff );
$str = mb_encode_numericentity($str, $convmap, "sjis-win");
```

See also: mb_decode_numericentity().

mb_ereg (PHP 4 >= 4.2.0)

Regular expression match with multibyte support

int **mb_ereg** (string pattern, string string [, array regs]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg() executes the regular expression match with multibyte support, and returns 1 if matches are found. If the optional third parameter was specified, the function returns the byte length of matched part, and therarray *regs* will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, FALSE will be returned.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_eregi()

mb_ereg_match (PHP 4 >= 4.2.0)

Regular expression match for multibyte string

bool **mb_ereg_match** (string pattern, string string [, string option]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_match() returns TRUE if *string* matches regular expression *pattern*, FALSE if not.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_ereg_replace (PHP 4 >= 4.2.0)

Replace regular expression with multibyte support

string **mb_ereg_replace** (string pattern, string replacement, string string [, array option]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_replace() scans *string* for matches to *pattern*, then replaces the matched text with *replacement* and returns the result string or FALSE on error. Multibyte character can be used in *pattern*.

Matching condition can be set by *option* parameter. If `i` is specified for this parameter, the case will be ignored. If `x` is specified, white space will be ignored. If `m` is specified, match will be executed in multiline mode and line break will be included in '.'. If `p` is specified, match will be executed in POSIX mode, line break will be considered as normal character. If `e` is specified, *replacement* string will be evaluated as PHP expression.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_eregi_replace().

mb_ereg_search (PHP 4 >= 4.2.0)

Multibyte regular expression match for predefined multibyte string

bool **mb_ereg_search** ([string pattern [, string option]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_search() returns TRUE if the multibyte string matches with the regular expression, FALSE for otherwise. The string for matching is set by mb_ereg_search_init(). If *pattern* is not specified, the previous one is used.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_getpos (PHP 4 >= 4.2.0)

Returns start point for next regular expression match

array **mb_ereg_search_getpos** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_search_getpos() returns the point to start regular expression match for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). The position is represented by bytes from the head of string.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_setpos().

mb_ereg_search_getregs (PHP 4 >= 4.2.0)

Retrive the result from the last multibyte regular expression match

array **mb_ereg_search_getregs** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_search_getregs() returns an array including the sub-string of matched part by last mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). If there are some maches, the first element will have the matched sub-string, the second element will have the first part grouped with brackets, the third element will have the second part grouped with brackets, and so on. It returns `FALSE` on error;

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_init (PHP 4 >= 4.2.0)

Setup string and regular expression for multibyte regular expression match

array **mb_ereg_search_init** (string string [, string pattern [, string option]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_search_init() sets *`string`* and *`pattern`* for multibyte regular expression. These values are used for mb_ereg_search(), mb_ereg_search_pos(), mb_ereg_search_regs(). It returns `TRUE` for success, `FALSE` for error.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_regs().

mb_ereg_search_pos (PHP 4 >= 4.2.0)

Return position and length of matched part of multibyte regular expression for predefined multibyte string

array **mb_ereg_search_pos** ([string pattern [, string option]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_search_pos() returns an array including position of matched part for multibyte regular expression. The first element of the array will be the beggining of matched part, the second element will be length (bytes) of matched part. It returns FALSE on error.

The string for match is specified by mb_ereg_search_init(). It it is not specified, the previous one will be used.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_regs (PHP 4 >= 4.2.0)

Returns the matched part of multibyte regular expression

array **mb_ereg_search_regs** ([string pattern [, string option]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_search_regs() executes the multibyte regular expression match, and if there are some matched part, it returns an array including substring of matched part as first element, the first

grouped part with brackets as second element, the second grouped part as third element, and so on. It returns FALSE on error.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_ereg_search_setpos (PHP 4 >= 4.2.0)

Set start point of next regular expression match

array **mb_ereg_search_setpos** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_search_setpos() sets the starting point of match for mb_ereg_search().

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_search_init().

mb_eregi (PHP 4 >= 4.2.0)

Regular expression match ignoring case with multibyte support

int **mb_eregi** (string pattern, string string [, array regs]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_eregi() executes the regular expression match with multibyte support, and returns 1 if matches are found. This function ignore case. If the optional third parameter was specified, the function

returns the byte length of matched part, and therarray *regs* will contain the substring of matched string. The functions returns 1 if it matches with the empty string. It no matche found or error happend, FALSE will be returned.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_eregi_replace (PHP 4 >= 4.2.0)

Replace regular expression with multibyte support ignoring case

string **mb_eregi_replace** (string pattern, string replace, string string) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_ereg_replace() scans *string* for matches to *pattern*, then replaces the matched text with *replacement* and returns the result string or FALSE on error. Multibyte character can be used in *pattern*. The case will be ignored.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

Nota: This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg_replace().

mb_get_info (PHP 4 >= 4.2.0)

Get internal settings of mbstring

string **mb_get_info** ([string type]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_get_info() returns internal setting parameter of mbstring.

If `type` isn't specified or is specified to "all", an array having the elements "internal_encoding", "http_output", "http_input", "func_overload" will be returned.

If `type` is specified for "http_output", "http_input", "internal_encoding", "func_overload", the specified setting parameter will be returned.

See also mb_internal_encoding(), mb_http_output().

mb_http_input (PHP 4 >= 4.0.6)

Detect HTTP input character encoding

string **mb_http_input** ([string type]) \linebreak

mb_http_input() returns result of HTTP input character encoding detection.

`type`: Input string specifies input type. "G" for GET, "P" for POST, "C" for COOKIE. If type is omitted, it returns last input type processed.

Return Value: Character encoding name. If **mb_http_input()** does not process specified HTTP input, it returns `FALSE`.

See also mb_internal_encoding(), mb_http_output(), mb_detect_order().

mb_http_output (PHP 4 >= 4.0.6)

Set/Get HTTP output character encoding

string **mb_http_output** ([string encoding]) \linebreak

If `encoding` is set, **mb_http_output()** sets HTTP output character encoding to `encoding`. Output after this function is converted to `encoding`. **mb_http_output()** returns `TRUE` for success and `FALSE` for failure.

If `encoding` is omitted, **mb_http_output()** returns current HTTP output character encoding.

See also mb_internal_encoding(), mb_http_input(), mb_detect_order().

mb_internal_encoding (PHP 4 >= 4.0.6)

Set/Get internal character encoding

string **mb_internal_encoding** ([string encoding]) \linebreak

mb_internal_encoding() sets internal character encoding to `encoding` If parameter is omitted, it returns current internal encoding.

`encoding` is used for HTTP input character encoding conversion, HTTP output character encoding conversion and default character encoding for string functions defined by mbstring module.

`encoding`: Character encoding name

Return Value: If *encoding* is set,**mb_internal_encoding()** returns TRUE for success, otherwise returns FALSE. If *encoding* is omitted, it returns current character encoding name.

Esempio 1. mb_internal_encoding() example

```
/* Set internal character encoding to UTF-8 */
mb_internal_encoding("UTF-8");

/* Display current internal character encoding */
echo mb_internal_encoding();
```

See also mb_http_input(), mb_http_output(), mb_detect_order().

mb_language (PHP 4 >= 4.0.6)

Set/Get current language

string **mb_language** ([string language]) \linebreak

mb_language() sets language. If *language* is omitted, it returns current language as string.

language setting is used for encoding e-mail messages. Valid languages are "Japanese", "ja","English","en" and "uni" (UTF-8). mb_send_mail() uses this setting to encode e-mail.

Language and its setting is ISO-2022-JP/Base64 for Japanese, UTF-8/Base64 for uni, ISO-8859-1/quoted printable for English.

Return Value: If *language* is set and *language* is valid, it returns TRUE. Otherwise, it returns FALSE. When *language* is omitted, it returns language name as string. If no language is set previously, it returns FALSE.

See also mb_send_mail().

mb_output_handler (PHP 4 >= 4.0.6)

Callback function converts character encoding in output buffer

string **mb_output_handler** (string contents, int status) \linebreak

mb_output_handler() is ob_start() callback function. **mb_output_handler()** converts characters in output buffer from internal character encoding to HTTP output character encoding.

4.1.0 or later version, this hanlder adds charset HTTP header when following conditions are met:

- Does not set `Content-Type` by header()
- Default MIME type begins with `text/`
- `http_output` setting is other than pass

`contents` : Output buffer contents

`status` : Output buffer status

Return Value: String converted

Esempio 1. mb_output_handler() example

```
mb_http_output("UTF-8");
ob_start("mb_output_handler");
```

> **Nota:** If you want to output some binary data such as image from PHP script, you must set output encoding to "pass" using mb_http_output().

See also ob_start().

mb_parse_str (PHP 4 >= 4.0.6)

Parse GET/POST/COOKIE data and set global variable

boolean **mb_parse_str** (string encoded_string [, array result]) \linebreak

mb_parse_str() parses GET/POST/COOKIE data and sets global variables. Since PHP does not provide raw POST/COOKIE data, it can only used for GET data for now. It preses URL encoded data, detects encoding, converts coding to internal encoding and set values to `result` array or global variables.

`encoded_string`: URL encoded data.

`result`: Array contains decoded and character encoding converted values.

Return Value: It returns `TRUE` for success or `FALSE` for failure.

See also mb_detect_order(), mb_internal_encoding().

mb_preferred_mime_name (PHP 4 >= 4.0.6)

Get MIME charset string

string **mb_preferred_mime_name** (string encoding) \linebreak

mb_preferred_mime_name() returns MIME `charset` string for character encoding `encoding`. It returns `charset` string.

Esempio 1. mb_preferred_mime_string() example

```
$outputenc = "sjis-win";
mb_http_output($outputenc);
ob_start("mb_output_handler");
header("Content-Type: text/html; charset=" . mb_preferred_mime_name($outputenc));
```

mb_regex_encoding (PHP 4 >= 4.2.0)

Returns current encoding for multibyte regex as string

string **mb_regex_encoding** ([string encoding]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_regex_encoding() returns the character encoding used by multibyte regex functions.

If the optional parameter *encoding* is specified, it is set to the character encoding for multibyte regex. The default value is the internal character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_internal_encoding(), mb_ereg()

mb_send_mail (PHP 4 >= 4.0.6)

Send encoded mail.

boolean **mb_send_mail** (string to, string subject, string message [, string additional_headers [, string additional_parameter]]) \linebreak

mb_send_mail() sends email. Headers and message are converted and encoded according to mb_language() setting. **mb_send_mail()** is wrapper function of mail(). See mail() for details.

to is mail addresses send to. Multiple recipients can be specified by putting a comma between each address in to. This parameter is not automatically encoded.

subject is subject of mail.

message is mail message.

additional_headers is inserted at the end of the header. This is typically used to add extra headers. Multiple extra headers are separated with a newline ("\n").

additional_parameter is a MTA command line parameter. It is useful when setting the correct Return-Path header when using sendmail.

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

See also mail(), mb_encode_mimeheader(), and mb_language().

mb_split (PHP 4 >= 4.2.0)

Split multibyte string using regular expression

array **mb_split** (string pattern, string string [, int limit]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

mb_split() split multibyte *string* using regular expression *pattern* and returns the result as an array.

If optional parameter *limit* is specified, it will be split in *limit* elements as maximum.

The internal encoding or the character encoding specified in mb_regex_encoding() will be used as character encoding.

> **Nota:** This function is supported in PHP 4.2.0 or higher.

See also: mb_regex_encoding(), mb_ereg().

mb_strcut (PHP 4 >= 4.0.6)

Get part of string

string **mb_strcut** (string str, int start [, int length [, string encoding]]) \linebreak

mb_strcut() returns the portion of *str* specified by the *start* and *length* parameters.

mb_strcut() performs equivalent operation as mb_substr() with different method. If *start* position is multi-byte character's second byte or larger, it starts from first byte of multi-byte character.

It subtracts string from *str* that is shorter than *length* AND character that is not part of multi-byte string or not being middle of shift sequence.

encoding is character encoding. If it is not set, internal character encoding is used.

See also mb_substr(), mb_internal_encoding().

mb_strimwidth (PHP 4 >= 4.0.6)

Get truncated string with specified width

string **mb_strimwidth** (string str, int start, int width, string trimmarker [, string encoding]) \linebreak

mb_strimwidth() truncates string *str* to specified *width*. It returns truncated string.

If *trimmarker* is set, *trimmarker* is appended to return value.

start is start position offset. Number of characters from the beginning of string. (First character is 0)

trimmarker is string that is added to the end of string when string is truncated.

encoding is character encoding. If it is omitted, internal encoding is used.

Esempio 1. mb_strimwidth() example

```
$str = mb_strimwidth($str, 0, 40, "..>");
```

See also: mb_strwidth(), mb_internal_encoding().

mb_strlen (PHP 4 >= 4.0.6)

Get string length

string **mb_strlen** (string str [, string encoding]) \linebreak

mb_strlen() returns number of characters in string *str* having character encoding *encoding*. A multi-byte character is counted as 1.

encoding is character encoding for *str*. If *encoding* is omitted, internal character encoding is used.

See also mb_internal_encoding(), strlen().

mb_strpos (PHP 4 >= 4.0.6)

Find position of first occurrence of string in a string

int **mb_strpos** (string haystack, string needle [, int offset [, string encoding]]) \linebreak

mb_strpos() returns the numeric position of the first occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns FALSE.

mb_strpos() performs multi-byte safe strpos() operation based on number of characters. *needle* position is counted from the beginning of the *haystack*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal character encoding is used. mb_strrpos() accepts `string` for *needle* where strrpos() accepts only character.

offset is search offset. If it is not specified, 0 is used.

encoding is character encoding name. If it is omitted, internal character encoding is used.

See also **mb_strpos()**, mb_internal_encoding(), strpos()

mb_strrpos (PHP 4 >= 4.0.6)

Find position of last occurrence of a string in a string

int **mb_strrpos** (string haystack, string needle [, string encoding]) \linebreak

mb_strrpos() returns the numeric position of the last occurrence of *needle* in the *haystack* string. If *needle* is not found, it returns `FALSE`.

mb_strrpos() performs multi-byte safe strrpos() operation based on number of characters. *needle* position is counted from the beginning of *haystack*. First character's position is 0. Second character position is 1.

If *encoding* is omitted, internal encoding is assumed. **mb_strrpos()** accepts `string` for *needle* where strrpos() accepts only character.

encoding is character encoding. If it is not specified, internal character encoding is used.

See also mb_strpos(), mb_internal_encoding(), strrpos().

mb_strwidth (PHP 4 >= 4.0.6)

Return width of string

int **mb_strwidth** (string str [, string encoding]) \linebreak

mb_strwidth() returns width of string *str*.

Multi-byte character usually twice of width compare to single byte character.

```
Character width

U+0000 - U+0019   0
U+0020 - U+1FFF   1
U+2000 - U+FF60   2
U+FF61 - U+FF9F   1
U+FFA0 -          2
```

encoding is character encoding. If it is omitted, internal encoding is used.

See also: mb_strimwidth(), mb_internal_encoding().

mb_substitute_character (PHP 4 >= 4.0.6)

Set/Get substitution character

mixed **mb_substitute_character** ([mixed substrchar]) \linebreak

mb_substitute_character() specifies substitution character when input character encoding is invalid or character code is not exist in output character encoding. Invalid characters may be substituted `NULL`(no output), string or integer value (Unicode character code value).

This setting affects mb_detect_encoding() and mb_send_mail().

substchar : Specify Unicode value as integer or specify as string as follows

- "none" : no output
- "long" : Output character code value (Example: U+3000,JIS+7E7E)

Return Value: If *substchar* is set, it returns `TRUE` for success, otherwise returns `FALSE`. If *substchar* is not set, it returns Unicode value or "`none`"/"`long`".

Esempio 1. mb_substitute_character() example

```
/* Set with Unicode U+3013 (GETA MARK) */
mb_substitute_character(0x3013);

/* Set hex format */
mb_substitute_character("long");

/* Display current setting */
echo mb_substitute_character();
```

mb_substr (PHP 4 >= 4.0.6)

Get part of string

string **mb_substr** (string str, int start [, int length [, string encoding]]) \linebreak

mb_substr() returns the portion of *str* specified by the *start* and *length* parameters.

mb_substr() performs multi-byte safe substr() operation based on number of characters. Position is counted from the beginning of *str*. First character's position is 0. Second character position is 1, and so on.

If *encoding* is omitted, internal encoding is assumed.

encoding is character encoding. If it is omitted, internal character encoding is used.

See also mb_strcut(), mb_internal_encoding().

LIV. MCAL functions

MCAL stands for Modular Calendar Access Library.

Libmcal is a C library for accessing calendars. It's written to be very modular, with pluggable drivers. MCAL is the calendar equivalent of the IMAP module for mailboxes.

With mcal support, a calendar stream can be opened much like the mailbox stream with the IMAP support. Calendars can be local file stores, remote ICAP servers, or other formats that are supported by the mcal library.

Calendar events can be pulled up, queried, and stored. There is also support for calendar triggers (alarms) and recurring events.

With libmcal, central calendar servers can be accessed, removing the need for any specific database or local file programming.

To get these functions to work, you have to compile PHP with `--with-mcal`. That requires the mcal

library to be installed. Grab the latest version from http://mcal.chek.com/ and compile and install it.

The following constants are defined when using the MCAL module. For weekdays :

- MCAL_SUNDAY
- MCAL_MONDAY
- MCAL_TUESDAY
- MCAL_WEDNESDAY
- MCAL_THURSDAY
- MCAL_FRIDAY
- MCAL_SATURDAY

For recurrence :

- MCAL_RECUR_NONE
- MCAL_RECUR_DAILY
- MCAL_RECUR_WEEKLY
- MCAL_RECUR_MONTHLY_MDAY
- MCAL_RECUR_MONTHLY_WDAY
- MCAL_RECUR_YEARLY

For months :

- MCAL_JANUARY
- MCAL_FEBRUARY
- MCAL_MARCH
- MCAL_APRIL
- MCAL_MAY
- MCAL_JUNE
- MCAL_JULY
- MCAL_AUGUST
- MCAL_SEPTEMBER
- MCAL_OCTOBER
- MCAL_NOVEMBER
- MCAL_DECEMBER

Most of the functions use an internal event structure that is unique for each stream. This alleviates the need to pass around large objects between functions. There are convenience functions for setting, initializing, and retrieving the event structure values.

mcal_append_event (PHP 4 >= 4.0.0)

Store a new event into an MCAL calendar

int **mcal_append_event** (int mcal_stream) \linebreak

mcal_append_event() Stores the global event into an MCAL calendar for the given stream.

Returns the id of the newly inserted event.

mcal_close (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Close an MCAL stream

int **mcal_close** (int mcal_stream, int flags) \linebreak

Closes the given mcal stream.

mcal_create_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Create a new MCAL calendar

string **mcal_create_calendar** (int stream, string calendar) \linebreak

Creates a new calendar named *`calendar`*.

mcal_date_compare (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Compares two dates

int **mcal_date_compare** (int a_year, int a_month, int a_day, int b_year, int b_month, int b_day) \linebreak

mcal_date_compare() Compares the two given dates, returns <0, 0, >0 if a<b, a==b, a>b respectively.

mcal_date_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns `TRUE` if the given year, month, day is a valid date

int **mcal_date_valid** (int year, int month, int day) \linebreak

mcal_date_valid() Returns `TRUE` if the given year, month and day is a valid date, `FALSE` if not.

mcal_day_of_week (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns the day of the week of the given date

int **mcal_day_of_week** (int year, int month, int day) \linebreak

mcal_day_of_week() returns the day of the week of the given date. Possible return values range from 0 for Sunday through 6 for Saturday.

mcal_day_of_year (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns the day of the year of the given date

int **mcal_** (int year, int month, int day) \linebreak

mcal_day_of_year() returns the day of the year of the given date.

mcal_days_in_month (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns the number of days in the given month

int **mcal_days_in_month** (int month, int leap year) \linebreak

mcal_days_in_month() Returns the number of days in the given month, taking into account if the given year is a leap year or not.

mcal_delete_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Delete an MCAL calendar

string **mcal_delete_calendar** (int stream, string calendar) \linebreak

Deletes the calendar named *`calendar`*.

mcal_delete_event (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Delete an event from an MCAL calendar

int **mcal_delete_event** (int mcal_stream [, int event_id]) \linebreak

mcal_delete_event() deletes the calendar event specified by the event_id.

Returns `TRUE`.

mcal_event_add_attribute (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Adds an attribute and a value to the streams global event structure

void **mcal_event_add_attribute** (int stream, string attribute, string value) \linebreak

mcal_event_add_attribute() adds an attribute to the stream's global event structure with the value given by "value".

mcal_event_init (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Initializes a streams global event structure

int **mcal_event_init** (int stream) \linebreak

mcal_event_init() initializes a streams global event structure. this effectively sets all elements of the structure to 0, or the default settings.

Returns TRUE.

mcal_event_set_alarm (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the alarm of the streams global event structure

int **mcal_event_set_alarm** (int stream, int alarm) \linebreak

mcal_event_set_alarm() sets the streams global event structure's alarm to the given minutes before the event.

Returns TRUE.

mcal_event_set_category (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the category of the streams global event structure

int **mcal_event_set_category** (int stream, string category) \linebreak

mcal_event_set_category() sets the streams global event structure's category to the given string.

Returns TRUE.

mcal_event_set_class (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the class of the streams global event structure

int **mcal_event_set_class** (int stream, int class) \linebreak

mcal_event_set_class() sets the streams global event structure's class to the given value. The class is either 1 for public, or 0 for private.

Returns TRUE.

mcal_event_set_description (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the description of the streams global event structure

int **mcal_event_set_description** (int stream, string description) \linebreak

mcal_event_set_description() sets the streams global event structure's description to the given string.

Returns TRUE.

mcal_event_set_end (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the end date and time of the streams global event structure

int **mcal_event_set_end** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]]) \linebreak

mcal_event_set_end() sets the streams global event structure's end date and time to the given values.

Returns TRUE.

mcal_event_set_recur_daily (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_daily** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_daily() sets the streams global event structure's recurrence to the given value to be reoccuring on a daily basis, ending at the given date.

mcal_event_set_recur_monthly_mday (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_monthly_mday** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_monthly_mday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by month day basis, ending at the given date.

mcal_event_set_recur_monthly_wday (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_monthly_wday** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_monthly_wday() sets the streams global event structure's recurrence to the given value to be reoccuring on a monthly by week basis, ending at the given date.

mcal_event_set_recur_none (PHP 3>= 3.0.15, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_none** (int stream) \linebreak

mcal_event_set_recur_none() sets the streams global event structure to not recur (event->recur_type is set to `MCAL_RECUR_NONE`).

mcal_event_set_recur_weekly (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_weekly** (int stream, int year, int month, int day, int interval, int weekdays) \line-break

mcal_event_set_recur_weekly() sets the streams global event structure's recurrence to the given value to be reoccuring on a weekly basis, ending at the given date.

mcal_event_set_recur_yearly (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the recurrence of the streams global event structure

int **mcal_event_set_recur_yearly** (int stream, int year, int month, int day, int interval) \linebreak

mcal_event_set_recur_yearly() sets the streams global event structure's recurrence to the given value to be reoccuring on a yearly basis,ending at the given date.

mcal_event_set_start (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the start date and time of the streams global event structure

int **mcal_event_set_start** (int stream, int year, int month [, int day [, int hour [, int min [, int sec]]]]) \linebreak

mcal_event_set_start() sets the streams global event structure's start date and time to the given values.

Returns TRUE.

mcal_event_set_title (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Sets the title of the streams global event structure

int **mcal_event_set_title** (int stream, string title) \linebreak

mcal_event_set_title() sets the streams global event structure's title to the given string.

Returns TRUE.

mcal_expunge (unknown)

Deletes all events marked for being expunged.

int **mcal_expunge** (int stream) \linebreak

mcal_expunge() Deletes all events which have been previously marked for deletion.

mcal_fetch_current_stream_event (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns an object containing the current streams event structure

object **mcal_fetch_current_stream_event** (int stream) \linebreak

mcal_fetch_current_stream_event() returns the current stream's event structure as an object containing:

- int id - ID of that event.
- int public - TRUE if the event if public, FALSE if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year

- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

mcal_fetch_event (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Fetches an event from the calendar stream

object **mcal_fetch_event** (int mcal_stream, int event_id [, int options]) \linebreak

mcal_fetch_event() fetches an event from the calendar stream specified by *`id`*.

Returns an event object consisting of:

- int id - ID of that event.
- int public - `TRUE` if the event if public, `FALSE` if it is private.
- string category - Category string of the event.
- string title - Title string of the event.
- string description - Description string of the event.
- int alarm - number of minutes before the event to send an alarm/reminder.
- object start - Object containing a datetime entry.
- object end - Object containing a datetime entry.
- int recur_type - recurrence type
- int recur_interval - recurrence interval
- datetime recur_enddate - recurrence end date
- int recur_data - recurrence data

All datetime entries consist of an object that contains:

- int year - year
- int month - month
- int mday - day of month
- int hour - hour
- int min - minutes
- int sec - seconds
- int alarm - minutes before event to send an alarm

The possible values for recur_type are:

- 0 - Indicates that this event does not recur
- 1 - This event recurs daily
- 2 - This event recurs on a weekly basis
- 3 - This event recurs monthly on a specific day of the month (e.g. the 10th of the month)
- 4 - This event recurs monthly on a sequenced day of the week (e.g. the 3rd Saturday)
- 5 - This event recurs on an annual basis

mcal_is_leap_year (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns if the given year is a leap year or not

int **mcal_is_leap_year** (int year) \linebreak

mcal_is_leap_year() returns 1 if the given year is a leap year, 0 if not.

mcal_list_alarms (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Return a list of events that has an alarm triggered at the given datetime

array **mcal_list_alarms** (int mcal_stream [, int begin_year [, int begin_month [, int begin_day [, int end_year [, int end_month [, int end_day]]]]]]) \linebreak

Returns an array of event ID's that has an alarm going off between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an optional beginning date and an end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_list_events (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Return a list of IDs for a date or a range of dates.

array **mcal_list_events** (int mcal_stream, object begin_date [, object end_date]) \linebreak

Returns an array of ID's that are between the start and end dates, or if just a stream is given, uses the start and end dates in the global event structure.

mcal_list_events() function takes in an beginning date and an optional end date for a calendar stream. An array of event id's that are between the given dates or the internal event dates are returned.

mcal_next_recurrence (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns the next recurrence of the event

int **mcal_next_recurrence** (int stream, int weekstart, array next) \linebreak

mcal_next_recurrence() returns an object filled with the next date the event occurs, on or after the supplied date. Returns empty date field if event does not occur or something is invalid. Uses weekstart to determine what day is considered the beginning of the week.

mcal_open (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Opens up an MCAL connection

int **mcal_open** (string calendar, string username, string password [, int options]) \linebreak

Returns an MCAL stream on success, FALSE on error.

mcal_open() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_popen (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Opens up a persistent MCAL connection

int **mcal_popen** (string calendar, string username, string password [, int options]) \linebreak

Returns an MCAL stream on success, FALSE on error.

mcal_popen() opens up an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also. The streams internal event structure is also initialized upon connection.

mcal_rename_calendar (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Rename an MCAL calendar

string **mcal_rename_calendar** (int stream, string old_name, string new_name) \linebreak

Renames the calendar *old_name* to *new_name*.

mcal_reopen (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Reopens an MCAL connection

int **mcal_reopen** (string calendar [, int options]) \linebreak

Reopens an MCAL stream to a new calendar.

mcal_reopen() reopens an MCAL connection to the specified *calendar* store. If the optional *options* is specified, passes the *options* to that mailbox also.

mcal_snooze (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Turn off an alarm for an event

int **mcal_snooze** (int id) \linebreak

mcal_snooze() turns off an alarm for a calendar event specified by the id.

Returns TRUE.

mcal_store_event (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Modify an existing event in an MCAL calendar

int **mcal_store_event** (int mcal_stream) \linebreak

mcal_store_event() Stores the modifications to the current global event for the given stream.

Returns the event id of the modified event on success and FALSE on error.

mcal_time_valid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Returns TRUE if the given year, month, day is a valid time

int **mcal_time_valid** (int hour, int minutes, int seconds) \linebreak

mcal_time_valid() Returns TRUE if the given hour, minutes and seconds is a valid time, FALSE if not.

mcal_week_of_year (PHP 4 >= 4.0.0)

Returns the week number of the given date

int **mcal_week_of_year** (int day, int month, int year) \linebreak

LV. Mcrypt Encryption Functions

This is an interface to the mcrypt library, which supports a wide variety of block algorithms such as DES, TripleDES, Blowfish (default), 3-WAY, SAFER-SK64, SAFER-SK128, TWOFISH, TEA, RC2 and GOST in CBC, OFB, CFB and ECB cipher modes. Additionally, it supports RC6 and IDEA which are considered "non-free".

Mcrypt can be used to encrypt and decrypt using the above mentioned ciphers. If you linked against libmcrypt-2.2.x, the four important mcrypt commands (mcrypt_cfb(), mcrypt_cbc(), mcrypt_ecb(), and mcrypt_ofb()) can operate in both modes which are named MCRYPT_ENCRYPT and MCRYPT_DECRYPT, respectively.

Esempio 1. Encrypt an input value with TripleDES under 2.2.x in ECB mode

```
<?php
$key = "this is a very secret key";
$input = "Let us meet at 9 o’clock at the secret place.";

$encrypted_data = mcrypt_ecb (MCRYPT_3DES, $key, $input, MCRYPT_ENCRYPT);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`.

If you linked against libmcrypt 2.4.x or 2.5.x, these functions are still available, but it is recommended that you use the advanced functions.

Esempio 2. Encrypt an input value with TripleDES under 2.4.x and higher in ECB mode

```
<?php
    $key = "this is a very secret key";
    $input = "Let us meet at 9 o’clock at the secret place.";

    $td = mcrypt_module_open (’tripledes’, ”, ’ecb’, ”);
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size ($td), MCRYPT_RAND);
    mcrypt_generic_init ($td, $key, $iv);
    $encrypted_data = mcrypt_generic ($td, $input);
    mcrypt_generic_end ($td);
?>
```

This example will give you the encrypted data as a string in `$encrypted_data`. For a full example see mcrypt_module_open().

Requirements

These functions work using mcrypt (http://mcrypt.hellug.gr/).

If you linked against libmcrypt 2.4.x or higher, the following additional block algorithms are supported: CAST, LOKI97, RIJNDAEL, SAFERPLUS, SERPENT and the following stream

ciphers: ENIGMA (crypt), PANAMA, RC4 and WAKE. With libmcrypt 2.4.x or higher another cipher mode is also available; nOFB.

Installation

To use it, download libmcrypt-x.x.tar.gz from here (http://mcrypt.hellug.gr/) and follow the included installation instructions. You need to compile PHP with the `--with-mcrypt` parameter to enable this extension. Make sure you compile libmcrypt with the option `--disable-posix-threads`.

Runtime Configuration

Resource types

Questa estensione non definisce alcun tipo di risorsa.

Predefined constants

Mcrypt can operate in four block cipher modes (CBC, OFB, CFB, and ECB). If linked against libmcrypt-2.4.x or higher the functions can also operate in the block cipher mode nOFB and in STREAM mode. Below you find a list with all supported encryption modes together with the constants that are defines for the encryption mode. For a more complete reference and discussion see Applied Cryptography by Schneier (ISBN 0-471-11709-9).

- MCRYPT_MODE_ECB (electronic codebook) is suitable for random data, such as encrypting other keys. Since data there is short and random, the disadvantages of ECB have a favorable negative effect.
- MCRYPT_MODE_CBC (cipher block chaining) is especially suitable for encrypting files where the security is increased over ECB significantly.
- MCRYPT_MODE_CFB (cipher feedback) is the best mode for encrypting byte streams where single bytes must be encrypted.
- MCRYPT_MODE_OFB (output feedback, in 8bit) is comparable to CFB, but can be used in applications where error propagation cannot be tolerated. It's insecure (because it operates in 8bit mode) so it is not recommended to use it.
- MCRYPT_MODE_NOFB (output feedback, in nbit) is comparable to OFB, but more secure because it operates on the block size of the algorithm.
- MCRYPT_MODE_STREAM is an extra mode to include some stream algorithms like WAKE or RC4.

Here is a list of ciphers which are currently supported by the mcrypt extension. For a complete list of supported ciphers, see the defines at the end of `mcrypt.h`. The general rule with the mcrypt-2.2.x API is that you can access the cipher from PHP with MCRYPT_ciphername. With the

libmcrypt-2.4.x and libmcrypt-2.5.x API these constants also work, but it is possible to specify the

name of the cipher as a string with a call to mcrypt_module_open().

- MCRYPT_3DES
- MCRYPT_ARCFOUR_IV (libmcrypt > 2.4.x only)
- MCRYPT_ARCFOUR (libmcrypt > 2.4.x only)
- MCRYPT_BLOWFISH
- MCRYPT_CAST_128
- MCRYPT_CAST_256
- MCRYPT_CRYPT
- MCRYPT_DES
- MCRYPT_DES_COMPAT (libmcrypt 2.2.x only)
- MCRYPT_ENIGMA (libmcrypt > 2.4.x only, alias for MCRYPT_CRYPT)
- MCRYPT_GOST
- MCRYPT_IDEA (non-free)
- MCRYPT_LOKI97 (libmcrypt > 2.4.x only)
- MCRYPT_MARS (libmcrypt > 2.4.x only, non-free)
- MCRYPT_PANAMA (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_128 (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_192 (libmcrypt > 2.4.x only)
- MCRYPT_RIJNDAEL_256 (libmcrypt > 2.4.x only)
- MCRYPT_RC2
- MCRYPT_RC4 (libmcrypt 2.2.x only)
- MCRYPT_RC6 (libmcrypt > 2.4.x only)
- MCRYPT_RC6_128 (libmcrypt 2.2.x only)
- MCRYPT_RC6_192 (libmcrypt 2.2.x only)
- MCRYPT_RC6_256 (libmcrypt 2.2.x only)
- MCRYPT_SAFER64
- MCRYPT_SAFER128
- MCRYPT_SAFERPLUS (libmcrypt > 2.4.x only)
- MCRYPT_SERPENT(libmcrypt > 2.4.x only)
- MCRYPT_SERPENT_128 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_192 (libmcrypt 2.2.x only)
- MCRYPT_SERPENT_256 (libmcrypt 2.2.x only)
- MCRYPT_SKIPJACK (libmcrypt > 2.4.x only)
- MCRYPT_TEAN (libmcrypt 2.2.x only)
- MCRYPT_THREEWAY
- MCRYPT_TRIPLEDES (libmcrypt > 2.4.x only)
- MCRYPT_TWOFISH (for older mcrypt 2.x versions, or mcrypt > 2.4.x)
- MCRYPT_TWOFISH128 (TWOFISHxxx are available in newer 2.x versions, but not in the 2.4.x

versions)

- MCRYPT_TWOFISH192
- MCRYPT_TWOFISH256
- MCRYPT_WAKE (libmcrypt > 2.4.x only)
- MCRYPT_XTEA (libmcrypt > 2.4.x only)

You must (in CFB and OFB mode) or can (in CBC mode) supply an initialization vector (IV) to the respective cipher function. The IV must be unique and must be the same when decrypting/encrypting. With data which is stored encrypted, you can take the output of a function of the index under which the data is stored (e.g. the MD5 key of the filename). Alternatively, you can transmit the IV together with the encrypted data (see chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic).

mcrypt_cbc (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Encrypt/decrypt data in CBC mode

string **mcrypt_cbc** (int cipher, string key, string data, int mode [, string iv]) \linebreak string **mcrypt_cbc** (string cipher, string key, string data, int mode [, string iv]) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_cfb (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Encrypt/decrypt data in CFB mode

string **mcrypt_cfb** (int cipher, string key, string data, int mode, string iv) \linebreak string **mcrypt_cfb** (string cipher, string key, string data, int mode [, string iv]) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_create_iv (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Create an initialization vector (IV) from a random source

string **mcrypt_create_iv** (int size, int source) \linebreak

mcrypt_create_iv() is used to create an IV.

mcrypt_create_iv() takes two arguments, *size* determines the size of the IV, *source* specifies the source of the IV.

The source can be MCRYPT_RAND (system random number generator), MCRYPT_DEV_RANDOM (read data from /dev/random) and MCRYPT_DEV_URANDOM (read data from /dev/urandom). If you use MCRYPT_RAND, make sure to call srand() before to initialize the random number generator.

Esempio 1. mcrypt_create_iv() example

```
<?php
    $size = mcrypt_get_iv_size (MCRYPT_CAST_256, MCRYPT_MODE_CFB);
    $iv = mcrypt_create_iv ($size, MCRYPT_DEV_RANDOM);
?>
```

The IV is only meant to give an alternative seed to the encryption routines. This IV does not need to be secret at all, though it can be desirable. You even can send it along with your ciphertext without loosing security.

More information can be found at http://www.ciphersbyritter.com/GLOSSARY.HTM#IV, http://fn2.freenet.edmonton.ab.ca/~jsavard/crypto/co0409.htm and in chapter 9.3 of Applied Cryptography by Schneier (ISBN 0-471-11709-9) for a discussion of this topic.

mcrypt_decrypt (PHP 4 >= 4.0.2)

Decrypts crypttext with given parameters

string **mcrypt_decrypt** (string cipher, string key, string data, string mode [, string iv]) \linebreak

mcrypt_decrypt() decrypts the data and returns the unencrypted data.

`Cipher` is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`Key` is the key with which the data is encrypted. If it's smaller that the required keysize, it is padded with '`\0`'.

`Data` is the data that will be decrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '`\0`'.

`Mode` is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *`IV`* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '`\0`'.

mcrypt_ecb (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Encrypt/decrypt data in ECB mode

string **mcrypt_ecb** (int cipher, string key, string data, int mode) \linebreak string **mcrypt_ecb** (string cipher, string key, string data, int mode [, string iv]) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mcrypt_enc_get_algorithms_name (PHP 4 >= 4.0.2)

Returns the name of the opened algorithm

string **mcrypt_enc_get_algorithms_name** (resource td) \linebreak

This function returns the name of the algorithm.

Esempio 1. mcrypt_enc_get_algorithms_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, ”, MCRYPT_MODE_CFB, ”);
    echo mcrypt_enc_get_algorithms_name($td). "\n";

    $td = mcrypt_module_open (’cast-256’, ”, MCRYPT_MODE_CFB, ”);
    echo mcrypt_enc_get_algorithms_name($td). "\n";
?>

Prints:
CAST-256
CAST-256
```

mcrypt_enc_get_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the opened algorithm

int **mcrypt_enc_get_block_size** (resource td) \linebreak

This function returns the block size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV of the opened algorithm

int **mcrypt_enc_get_iv_size** (resource td) \linebreak

This function returns the size of the iv of the algorithm specified by the encryption descriptor in bytes. If it returns ’0’ then the IV is ignored in the algorithm. An IV is used in cbc, cfb and ofb modes, and in some algorithms in stream mode.

mcrypt_enc_get_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

int **mcrypt_enc_get_key_size** (resource td) \linebreak

This function returns the maximum supported key size of the algorithm specified by the encryption descriptor *td* in bytes.

mcrypt_enc_get_modes_name (PHP 4 >= 4.0.2)

Returns the name of the opened mode

string **mcrypt_enc_get_modes_name** (resource td) \linebreak

This function returns the name of the mode.

Esempio 1. mcrypt_enc_get_modes_name() example

```
<?php
    $td = mcrypt_module_open (MCRYPT_CAST_256, ″, MCRYPT_MODE_CFB, ″);
    echo mcrypt_enc_get_modes_name($td). "\n";

    $td = mcrypt_module_open ('cast-256', ″, 'ecb', ″);
    echo mcrypt_enc_get_modes_name($td). "\n";
?>

Prints:
CFB
ECB
```

mcrypt_enc_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

array **mcrypt_enc_get_supported_key_sizes** (resource td) \linebreak

Returns an array with the key sizes supported by the algorithm specified by the encryption descriptor. If it returns an empty array then all key sizes between 1 and mcrypt_enc_get_key_size() are supported by the algorithm.

Esempio 1. mcrypt_enc_get_supported_key_sizes() example

```
<?php
    $td = mcrypt_module_open ('rijndael-256', ″, 'ecb', ″);
    var_dump (mcrypt_enc_get_supported_key_sizes($td));
?>

This will print:

array(3) {
  [0]=>
  int(16)
  [1]=>
  int(24)
  [2]=>
```

```
  int(32)
}
?>
```

mcrypt_enc_is_block_algorithm (PHP 4 >= 4.0.2)

Checks whether the algorithm of the opened mode is a block algorithm

bool **mcrypt_enc_is_block_algorithm** (resource td) \linebreak

This function returns `TRUE` if the algorithm is a block algorithm, or `FALSE` if it is a stream algorithm.

mcrypt_enc_is_block_algorithm_mode (PHP 4 >= 4.0.2)

Checks whether the encryption of the opened mode works on blocks

bool **mcrypt_enc_is_block_algorithm_mode** (resource td) \linebreak

This function returns `TRUE` if the mode is for use with block algorithms, otherwise it returns `FALSE`. (eg. `FALSE` for stream, and `TRUE` for cbc, cfb, ofb).

mcrypt_enc_is_block_mode (PHP 4 >= 4.0.2)

Checks whether the opened mode outputs blocks

bool **mcrypt_enc_is_block_mode** (resource td) \linebreak

This function returns `TRUE` if the mode outputs blocks of bytes or `FALSE` if it outputs bytes. (eg. `TRUE` for cbc and ecb, and `FALSE` for cfb and stream).

mcrypt_enc_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the opened module

bool **mcrypt_enc_self_test** (resource td) \linebreak

This function runs the self test on the algorithm specified by the descriptor *td*. If the self test succeeds it returns `FALSE`. In case of an error, it returns `TRUE`.

mcrypt_encrypt (PHP 4 >= 4.0.2)

Encrypts plaintext with given parameters

string **mcrypt_encrypt** (string cipher, string key, string data, string mode [, string iv]) \linebreak

mcrypt_encrypt() encrypts the data and returns the encrypted data.

Cipher is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

Key is the key with which the data will be encrypted. If it's smaller that the required keysize, it is padded with '\0'. It is better not to use ASCII strings for keys. It is recommended to use the mhash functions to create a key from a string.

Data is the data that will be encrypted with the given cipher and mode. If the size of the data is not n * blocksize, the data will be padded with '\0'. The returned crypttext can be larger that the size of the data that is given by *data*.

Mode is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

The *IV* parameter is used for the initialisation in CBC, CFB, OFB modes, and in some algorithms in STREAM mode. If you do not supply an IV, while it is needed for an algorithm, the function issues a warning and uses an IV with all bytes set to '\0'.

Esempio 1. mcrypt_encrypt() Example

```
<?php
    $iv = mcrypt_create_iv (mcrypt_get_iv_size (MCRYPT_RIJNDAEL_256, MCRYPT_MODE_ECB), MC
    $key = "This is a very secret key";
    $text = "Meet me at 11 o'clock behind the monument.";
    echo strlen ($text)."\n";

    $crypttext = mcrypt_encrypt (MCRYPT_RIJNDAEL_256, $key, $text, MCRYPT_MODE_ECB, $iv);
    echo strlen ($crypttext)."\n";
?>
```

The above example will print out:

```
42
64
```

See also mcrypt_module_open() for a more advanced API and an example.

mcrypt_generic (PHP 4 >= 4.0.2)

This function encrypts data

string **mcrypt_generic** (resource td, string data) \linebreak

This function encrypts data. The data is padded with "`\0`" to make sure the length of the data is n * blocksize. This function returns the encrypted data. Note that the length of the returned string can in fact be longer then the input, due to the padding of the data.

The encryption handle should alwayws be initialized with mcrypt_generic_init() with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling mcrypt_generic_deinit(). See mcrypt_module_open() for an example.

See also mdecrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

mcrypt_generic_deinit (PHP 4 >= 4.1.1)

This function deinitializes an encryption module

bool **mcrypt_generic_deinit** (resource td) \linebreak

This function terminates encryption specified by the encryption descriptor (*`td`*). It clears all buffers, but does not close the module. You need to call mcrypt_module_close() yourself. (But PHP does this for you at the end of the script. Returns `FALSE` on error, or `TRUE` on succes.

See for an example mcrypt_module_open() and the entry on mcrypt_generic_init().

mcrypt_generic_end (PHP 4 >= 4.0.2)

This function terminates encryption

bool **mcrypt_generic_end** (resource td) \linebreak

This function is deprecated, use mcrypt_generic_deinit() instead. It can cause crashes when used with mcrypt_module_close() due to multiple buffer frees.

This function terminates encryption specified by the encryption descriptor (*`td`*). Actually it clears all buffers, and closes all the modules used. Returns `FALSE` on error, or `TRUE` on succes.

mcrypt_generic_init (PHP 4 >= 4.0.2)

This function initializes all buffers needed for encryption

int **mcrypt_generic_init** (resource td, string key, string iv) \linebreak

The maximum length of the key should be the one obtained by calling mcrypt_enc_get_key_size() and every value smaller than this is legal. The IV should normally have the size of the algorithms block size, but you must obtain the size by calling mcrypt_enc_get_iv_size(). IV is ignored in ECB. IV MUST exist in CFB, CBC, STREAM, nOFB and OFB modes. It needs to be random and unique (but not secret). The same IV must be used for encryption/decryption. If you do not want to use it you should set it to zeros, but this is not recommended. The function returns a negative value on error.

You need to call this function before every call to mcrypt_generic() or mdecrypt_generic().

See for an example mcrypt_module_open() and the entry on mcrypt_generic_deinit().

mcrypt_get_block_size (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Get the block size of the specified cipher

int **mcrypt_get_block_size** (int cipher) \linebreak int **mcrypt_get_block_size** (string cipher, string module) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or 2.5.x.

mcrypt_get_block_size() is used to get the size of a block of the specified `cipher` (in combination with an encryption mode).

This example shows how to use this function when linked against libmcrypt 2.4.x and 2.5.x.

Esempio 1. mcrypt_get_block_size() example

```
<?php
    echo mcrypt_get_block_size ('tripledes', 'ecb');
?>

Prints:
8
```

See also: mcrypt_get_key_size() and mcrypt_encrypt().

mcrypt_get_cipher_name (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Get the name of the specified cipher

string **mcrypt_get_cipher_name** (int cipher) \linebreak string **mcrypt_get_cipher_name** (string cipher) \linebreak

mcrypt_get_cipher_name() is used to get the name of the specified cipher.

mcrypt_get_cipher_name() takes the cipher number as an argument (libmcrypt 2.2.x) or takes the cipher name as an argument (libmcrypt 2.4.x or higher) and returns the name of the cipher or `FALSE`, if the cipher does not exist.

Esempio 1. mcrypt_get_cipher_name() Example

```
<?php
   $cipher = MCRYPT_TripleDES;
```

```
    echo mcrypt_get_cipher_name ($cipher);
?>
```

The above example will produce:

```
3DES
```

mcrypt_get_iv_size (PHP 4 >= 4.0.2)

Returns the size of the IV belonging to a specific cipher/mode combination

int **mcrypt_get_iv_size** (resource td) \linebreak int **mcrypt_get_iv_size** (string cipher, string mode) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

mcrypt_get_iv_size() returns the size of the Initialisation Vector (IV) in bytes. On error the function returns `FALSE`. If the IV is ignored in the specified cipher/mode combination zero is returned.

`cipher` is one of the MCRYPT_ciphername constants of the name of the algorithm as string.

`mode` is one of the MCRYPT_MODE_modename constants of one of "ecb", "cbc", "cfb", "ofb", "nofb" or "stream".

`td` is the resource that is returned by mcrypt_module_open().

Esempio 1. mcrypt_create_iv() example

```
<?php
    $size = mcrypt_get_iv_size (MCRYPT_CAST_256, MCRYPT_MODE_CFB);

    $size = mcrypt_get_iv_size ('des', 'ecb');
?>
```

See also: mcrypt_create_iv()

mcrypt_get_key_size (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Get the key size of the specified cipher

int **mcrypt_get_key_size** (int cipher) \linebreak int **mcrypt_get_key_size** (string cipher, string module) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or 2.5.x.

mcrypt_get_key_size() is used to get the size of a key of the specified *cipher* (in combination with an encryption mode).

This example shows how to use this function when linked against libmcrypt 2.4.x and 2.5.x.

Esempio 1. mcrypt_get_block_size() example

```
<?php
    echo mcrypt_get_key_size ('tripledes', 'ecb');
?>

Prints:
24
```

See also: mcrypt_get_block_size() and mcrypt_encrypt().

mcrypt_list_algorithms (PHP 4 >= 4.0.2)

Get an array of all supported ciphers

array **mcrypt_list_algorithms** ([string lib_dir]) \linebreak

mcrypt_list_algorithms() is used to get an array of all supported algorithms in the *lib_dir* parameter.

mcrypt_list_algorithms() takes an optional *lib_dir* parameter which specifies the directory where all algorithms are located. If not specifies, the value of the mcrypt.algorithms_dir `php.ini` directive is used.

Esempio 1. mcrypt_list_algorithms() Example

```
<?php
    $algorithms = mcrypt_list_algorithms ("/usr/local/lib/libmcrypt");

    foreach ($algorithms as $cipher) {
        echo "$cipher<br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the "/usr/local/lib/libmcrypt" directory.

mcrypt_list_modes (PHP 4 >= 4.0.2)

Get an array of all supported modes

array **mcrypt_list_modes** ([string lib_dir]) \linebreak

mcrypt_list_modes() is used to get an array of all supported modes in the *`lib_dir`*.

mcrypt_list_modes() takes as optional parameter a directory which specifies the directory where all modes are located. If not specifies, the value of the mcrypt.modes_dir `php.ini` directive is used.

Esempio 1. mcrypt_list_modes() Example

```
<?php
    $modes = mcrypt_list_modes ();

    foreach ($modes as $mode) {
        echo "$mode <br />\n";
    }
?>
```

The above example will produce a list with all supported algorithms in the default mode directory. If it is not set with the ini directive mcrypt.modes_dir, the default directory of mcrypt is used (which is /usr/local/lib/libmcrypt).

mcrypt_module_close (PHP 4 >= 4.0.2)

Close the mcrypt module

bool **mcrypt_module_close** (resource td) \linebreak

This function closes the specified encryption handle.

See mcrypt_module_open() for an example.

mcrypt_module_get_algo_block_size (PHP 4 >= 4.0.2)

Returns the blocksize of the specified algorithm

int **mcrypt_module_get_algo_block_size** (string algorithm [, string lib_dir]) \linebreak

This function returns the block size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_algo_key_size (PHP 4 >= 4.0.2)

Returns the maximum supported keysize of the opened mode

int **mcrypt_module_get_algo_key_size** (string algorithm [, string lib_dir]) \linebreak

This function returns the maximum supported key size of the algorithm specified in bytes. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_get_supported_key_sizes (PHP 4 >= 4.0.2)

Returns an array with the supported keysizes of the opened algorithm

array **mcrypt_module_get_supported_key_sizes** (string algorithm [, string lib_dir]) \linebreak

Returns an array with the key sizes supported by the specified algorithm. If it returns an empty array then all key sizes between 1 and mcrypt_module_get_algo_key_size() are supported by the algorithm. The optional *lib_dir* parameter can contain the location where the mode module is on the system.

See also mcrypt_enc_get_supported_key_sizes() which is used on open encryption modules.

mcrypt_module_is_block_algorithm (PHP 4 >= 4.0.2)

This function checks whether the specified algorithm is a block algorithm

bool **mcrypt_module_is_block_algorithm** (string algorithm [, string lib_dir]) \linebreak

This function returns `TRUE` if the specified algorithm is a block algorithm, or `FALSE` is it is a stream algorithm. The optional *lib_dir* parameter can contain the location where the algorithm module is on the system.

mcrypt_module_is_block_algorithm_mode (PHP 4 >= 4.0.2)

This function returns if the the specified module is a block algorithm or not

bool **mcrypt_module_is_block_algorithm_mode** (string mode [, string lib_dir]) \linebreak

This function returns `TRUE` if the mode is for use with block algorithms, otherwise it returns `FALSE`. (eg. `FALSE` for stream, and `TRUE` for cbc, cfb, ofb). The optional *lib_dir* parameter can contain the location where the mode module is on the system.

mcrypt_module_is_block_mode (PHP 4 >= 4.0.2)

This function returns if the the specified mode outputs blocks or not

bool **mcrypt_module_is_block_mode** (string mode [, string lib_dir]) \linebreak

This function returns `TRUE` if the mode outputs blocks of bytes or `FALSE` if it outputs just bytes. (eg. `TRUE` for cbc and ecb, and `FALSE` for cfb and stream). The optional *`lib_dir`* parameter can contain the location where the mode module is on the system.

mcrypt_module_open (PHP 4 >= 4.0.2)

This function opens the module of the algorithm and the mode to be used

resource **mcrypt_module_open** (string algorithm, string algorithm_directory, string mode, string mode_directory) \linebreak

This function opens the module of the algorithm and the mode to be used. The name of the algorithm is specified in algorithm, eg. "twofish" or is one of the MCRYPT_ciphername constants. The module is closed by calling mcrypt_module_close(). Normally it returns an encryption descriptor, or `FALSE` on error.

The *`algorithm_directory`* and *`mode_directory`* are used to locate the encryption modules. When you supply a directory name, it is used. When you set one of these to the empty string (""), the value set by the *`mcrypt.algorithms_dir`* or *`mcrypt.modes_dir`* ini-directive is used. When these are not set, the default directories that are used are the ones that were compiled in into libmcrypt (usally /usr/local/lib/libmcrypt).

Esempio 1. mcrypt_module_open() Example

```
<?php
    $td = mcrypt_module_open (MCRYPT_DES, ″, MCRYPT_MODE_ECB, ′/usr/lib/mcrypt-
modes′);
    $td = mcrypt_module_open (′rijndael-256′, ″, ′ofb′, ″);
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher an dmode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

Esempio 2. Using mcrypt_module_open() in encryption

```
<?php
    /* Open the cipher */
    $td = mcrypt_module_open (′rijndael-256′, ″, ′ofb′, ″);
```

```
    /* Create the IV and determine the keysize length */
    $iv = mcrypt_create_iv (mcrypt_enc_get_iv_size($td), MCRYPT_DEV_RANDOM);
    $ks = mcrypt_enc_get_key_size ($td);

    /* Create key */
    $key = substr (md5 ('very secret key'), 0, $ks);

    /* Intialize encryption */
    mcrypt_generic_init ($td, $key, $iv);

    /* Encrypt data */
    $encrypted = mcrypt_generic ($td, 'This is very important data');

    /* Terminate encryption handler */
    mcrypt_generic_deinit ($td);

    /* Initialize encryption module for decryption */
    mcrypt_generic_init ($td, $key, $iv);

    /* Decrypt encrypted string */
    $decrypted = mdecrypt_generic ($td, $encrypted);

    /* Terminate decryption handle and close module */
    mcrypt_generic_deinit ($td);
    mcrypt_module_close ($td);

    /* Show string */
    echo trim ($decrypted)."\n";
?>
```

The first line in the example above will try to open the DES cipher from the default directory and the EBC mode from the directory `/usr/lib/mcrypt-modes`. The second example uses strings as name for the cipher an dmode, this only works when the extension is linked against libmcrypt 2.4.x or 2.5.x.

See also mcrypt_module_close(), mcrypt_generic(), mdecrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

mcrypt_module_self_test (PHP 4 >= 4.0.2)

This function runs a self test on the specified module

bool **mcrypt_module_self_test** (string algorithm [, string lib_dir]) \linebreak

This function runs the self test on the algorithm specified. The optional *lib_dir* parameter can contain the location of where the algorithm module is on the system.

The function returns `TRUE` if the self test succeeds, or `FALSE` when if fails.

mcrypt_ofb (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Encrypt/decrypt data in OFB mode

string **mcrypt_ofb** (int cipher, string key, string data, int mode, string iv) \linebreak string **mcrypt_ofb** (string cipher, string key, string data, int mode [, string iv]) \linebreak

The first prototype is when linked against libmcrypt 2.2.x, the second when linked against libmcrypt 2.4.x or higher.

This function should not be used anymore, see mcrypt_generic() and mdecrypt_generic() for replacements.

mdecrypt_generic (PHP 4 >= 4.0.2)

This function decrypts data

string **mdecrypt_generic** (resource td, string data) \linebreak

This function decrypts data. Note that the length of the returned string can in fact be longer then the unencrypted string, due to the padding of the data.

Esempio 1. mdecrypt_generic() Example

```
<?php
    /* Data */
    $key = ’this is a very long key, even too long for the cipher’;
    $plain_text = ’very important data’;

    /* Open module, and create IV */
    $td = mcrypt_module_open (’des’, ”, ’ecb’, ”);
    $key = substr ($key, 0, mcrypt_enc_get_key_size ($td));
    $iv_size = mcrypt_enc_get_iv_size ($td);
    $iv = mcrypt_create_iv ($iv_size, MCRYPT_RAND);

    /* Initialize encryption handle */
    if (mcrypt_generic_init ($td, $key, $iv) != -1) {

        /* Encrypt data */
        $c_t = mcrypt_generic ($td, $plain_text);
        mcrypt_generic_deinit ($td);

        /* Reinitialize buffers for decryption */
        mcrypt_generic_init ($td, $key, $iv);
        $p_t = mdecrypt_generic ($td, $c_t);

        /* Clean up */
        mcrypt_generic_deinit ($td);
        mcrypt_module_close ($td);
    }

    if (strncmp ($p_t, $plain_text, strlen($plain_text)) == 0) {
        echo "ok\n";
```

```
    } else {
        echo "error\n";
    }
?>
```

The above example shows how to check if the data before the encryption is the same as the data after the decryption. It is very important to reinitialize the encryption buffer with mcrypt_generic_init() before you try to decrypt the data.

The decryption handle should alwayws be initialized with mcrypt_generic_init() with a key and an IV before calling this function. Where the encryption is done, you should free the encryption buffers by calling mcrypt_generic_deinit(). See mcrypt_module_open() for an example.

See also mcrypt_generic(), mcrypt_generic_init() and mcrypt_generic_deinit().

LVI. Mhash Functions

These functions are intended to work with mhash (http://mhash.sourceforge.net/).

This is an interface to the mhash library. mhash supports a wide variety of hash algorithms such as MD5, SHA1, GOST, and many others.

To use it, download the mhash distribution from its web site (http://mhash.sourceforge.net/) and follow the included installation instructions. You need to compile PHP with the `--with-mhash` parameter to enable this extension.

Mhash can be used to create checksums, message digests, message authentication codes, and more.

Esempio 1. Compute the MD5 digest and hmac and print it out as hex

```
<?php
$input = "what do ya want for nothing?";
$hash = mhash (MHASH_MD5, $input);
print "The hash is ".bin2hex ($hash)."<br />\n";
$hash = mhash (MHASH_MD5, $input, "Jefe");
print "The hmac is ".bin2hex ($hash)."<br />\n";
?>
```

This will produce:

```
The hash is d03cb659cbf9192dcd066272249f8412
The hmac is 750c783e6ab0b503eaa86e310a5db738
```

For a complete list of supported hashes, refer to the documentation of mhash. The general rule is that you can access the hash algorithm from PHP with MHASH_HASHNAME. For example, to access TIGER you use the PHP constant MHASH_TIGER.

Here is a list of hashes which are currently supported by mhash. If a hash is not listed here, but is

listed by mhash as supported, you can safely assume that this documentation is outdated.

- MHASH_MD5
- MHASH_SHA1
- MHASH_HAVAL256
- MHASH_HAVAL192
- MHASH_HAVAL160
- MHASH_HAVAL128
- MHASH_RIPEMD160
- MHASH_GOST
- MHASH_TIGER
- MHASH_CRC32
- MHASH_CRC32B

mhash (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Compute hash

string **mhash** (int hash, string data [, string key]) \linebreak

mhash() applies a hash function specified by *hash* to the *data* and returns the resulting hash (also called digest). If the *key* is specified it will return the resulting HMAC. HMAC is keyed hashing for message authentication, or simply a message digest that depends on the specified key. Not all algorithms supported in mhash can be used in HMAC mode. In case of an error returns FALSE.

mhash_count (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Get the highest available hash id

int **mhash_count** (void) \linebreak

mhash_count() returns the highest available hash id. Hashes are numbered from 0 to this hash id.

Esempio 1. Traversing all hashes

```
<?php

$nr = mhash_count();

for ($i = 0; $i <= $nr; $i++) {
    echo sprintf ("The blocksize of %s is %d\n",
        mhash_get_hash_name ($i),
        mhash_get_block_size ($i));
}
?>
```

mhash_get_block_size (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Get the block size of the specified hash

int **mhash_get_block_size** (int hash) \linebreak

mhash_get_block_size() is used to get the size of a block of the specified *hash*.

mhash_get_block_size() takes one argument, the *hash* and returns the size in bytes or FALSE, if the *hash* does not exist.

mhash_get_hash_name (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Get the name of the specified hash

string **mhash_get_hash_name** (int hash) \linebreak

mhash_get_hash_name() is used to get the name of the specified hash.

mhash_get_hash_name() takes the hash id as an argument and returns the name of the hash or `FALSE`, if the hash does not exist.

Esempio 1. mhash_get_hash_name() example

```
<?php
$hash = MHASH_MD5;

print mhash_get_hash_name ($hash);
?>
```

The above example will print out:

```
MD5
```

mhash_keygen_s2k (PHP 4 >= 4.0.4)

Generates a key

string **mhash_keygen_s2k** (int hash, string password, string salt, int bytes) \linebreak

mhash_keygen_s2k() generates a key that is *`bytes`* long, from a user given password. This is the Salted S2K algorithm as specified in the OpenPGP document (RFC 2440). That algorithm will use the specified *`hash`* algorithm to create the key. The *`salt`* must be different and random enough for every key you generate in order to create different keys. That salt must be known when you check the keys, thus it is a good idea to append the key to it. Salt has a fixed length of 8 bytes and will be padded with zeros if you supply less bytes.

Keep in mind that user supplied passwords are not really suitable to be used as keys in cryptographic algorithms, since users normally choose keys they can write on keyboard. These passwords use only 6 to 7 bits per character (or less). It is highly recommended to use some kind of tranformation (like this function) to the user supplied key.

LVII. Funzioni per Microsoft SQL Server

L'estensione per MSSQL è disponibile solamente sui sistemi Win32. Sulle altre piattaforme si può utilizzare il modulo Sybase per connettersi con database MSSQL.

Queste funzioni permettono di accedere a database MS SQL Server. Per potere funzionare è richiesto che sia installato il MS SQL Client Tools sullo stesso sistema su cui è il installato il PHP. Il Client Tools può essere installato o dal cd di MS SQL Server, o copiando il file ntwdblib.dll dalla directory \winnt\system32 del server alla directory \winnt\system32 della macchina su cui è installato il PHP. La copia del file ntwdblib.dll permette solo l'accesso al database. La configurazione del client richiede comunque l'installazione di tutto il pacchetto MS SQL Client Tools.

Il modulo MSSQL si abilita aggiungendo extension=php_mssql.dll al file di configurazione `php.ini`.

mssql_bind (PHP 4 >= 4.1.0)

Aggiunge un parametro ad una procedura memorizzata (stored procedure) locale o remota

int **mssql_bind** (int stmt, string param_name, mixed var, int type [, int is_output [, int is_null [, int maxlen]]]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mssql_close (PHP 3, PHP 4 >= 4.0.0)

Chiude la connessione con MS SQL Server

int **mssql_close** ([int id_connessione]) \linebreak

Restituisce: `TRUE` se l'operazione riesce, falso se si verifica un errore.

La funzione **mssql_close()** chiude la connessione ad un database MS SQL Server che è associata all' argomento id_connessione. Se l' id_connessione non viene indicato, si fa riferimento all'ultima connessione aperta.

Nota: solitamente non è necessario l'uso della funzione, dato che tutte le connessioni non-persistenti sono chiuse automaticamente al termine dell'esecuzione dello script.

mssql_close() non chiude i collegamenti persistenti aperti utilizzando mssql_pconnect().

Vedere anche: mssql_connect(), mssql_pconnect().

mssql_connect (PHP 3, PHP 4 >= 4.0.0)

Apre una connessione con un server MS SQL

int **mssql_connect** ([string nome_server [, string nome_utente [, string password]]]) \linebreak

Restituisce: un identificativo di connessione su l'operazione riesce, oppure falso se si verifica un errore.

La funzione **mssql_connect()** realizza una connessione con un server MS SQL. L' argomento nome_server deve essere un nome valido di server come definito nel file 'interfaces'.

Qualora la funzione **mssql_connect()** venga eseguita una seconda volta con i medesimi parametri, non viene realizzata una nuova connessione, ma, invece, viene restituito l'identificativo della connessione già aperta.

La connessione con il server verrà chiusa non appena lo script terminerà l'esecuzione, a meno che la connessione non sia già stata chiusa utilizzando la funzione mssql_close().

Vedere anche mssql_pconnect(), e mssql_close().

mssql_data_seek (PHP 3, PHP 4 >= 4.0.0)

Sposta il puntatore di riga interno

int **mssql_data_seek** (int id_risultato, int numero_riga) \linebreak

Restituisce: vero se l'operazione riesce, falso se si verifica un errore.

La funzione **mssql_data_seek()** sposta il puntatore di riga, interno al risultato associato all'identificativo di risultato, alla riga indicata dall'argomento numero_riga. La chiamata successiva a mssql_fetch_row() restituirà la riga richiesta.

Vedere anche **mssql_data_seek()**.

mssql_execute (PHP 4 >= 4.1.0)

Esegue una procedura memorizzata su un database MS-SQL

int **mssql_execute** (int stmt) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mssql_fetch_array (PHP 3, PHP 4 >= 4.0.0)

Restituisce una riga in un array

int **mssql_fetch_array** (int id_risultato) \linebreak

La funzione restituisce: un array corrispondente alla riga estratta, oppure falso se non vi sono più righe.

La funzione **mssql_fetch_array()** è un' estensione della funzione mssql_fetch_row(). Oltre a memorizzare i dati in un array con indice numerico, la funzione memorizza i dati in un array associativo in cui la chiave è costituita dal nome del campo.

Un aspetto da notare è che la funzione **mssql_fetch_array()** NON è significativamente più lenta rispetto a mssql_fetch_row(), mentre nel contempo fornisce funzionalità maggiori.

Per ulteriori dettagli vedere anche mssql_fetch_row().

mssql_fetch_assoc (PHP 4 >= 4.2.0)

Restituisce un array associativo con i dati di una riga dal set di risultati indicato da id_risultato

array **mssql_fetch_assoc** (int id_risultato [, int tipo_risultato]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mssql_fetch_batch (PHP 4 >= 4.0.4)

Restituisce il successivo gruppo di record

int **mssql_fetch_batch** (string indice_risultato) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mssql_fetch_field (PHP 3, PHP 4 >= 4.0.0)

Restituisce le informazioni di un campo

object **mssql_fetch_field** (int id_risultato [, int offset_campo]) \linebreak

La funzione restituisce un oggetto contenente le informazioni sul campo.

La funzione **mssql_fetch_field()** può essere utilizzata per ottenere informazioni sui campi presenti nel risultato di una certa query. Se non viene specificato l'argomento offset_campo, la funzione restituisce il campo successivo che non è ancora stato restituito da **mssql_fetch_field()**.

Le proprietà dell'oggetto sono:

- name - nome della colonna. Se la colonna è il risultato di una funzione, questa proprietà è valorizzata con "computed#N", dove #N è un numero progressivo.
- column_source - tabella da cui sono ricavate le colonne
- max_length - lunghezza massima della colonna
- numeric - 1 se la colonna è numerica

Vedere anche mssql_field_seek().

mssql_fetch_object (PHP 3, PHP 4 >= 4.0.0)

Restituisce una riga come oggetto

int **mssql_fetch_object** (int id_risultato) \linebreak

La funzione restituisce un oggetto le cui proprietà corrispondono alla riga estratta, oppure falso se non vi sono più righe.

La funzione **mssql_fetch_object()** è simile a mssql_fetch_array(), tranne che per una differenza, la prima restituisce un oggetto, la seconda un array. Indirettamente questo significa che si può accedere ai dati solo attraverso il nome dei campi e non tramite il loro offset (i numeri non sono dei validi nomi di proprietà).

A livello di velocità il comportamento è simile a mssql_fetch_array(), e quasi veloce come mssql_fetch_row() (la differenza è insignificante).

Vedere anche mssql_fetch-array() and mssql_fetch-row().

mssql_fetch_row (PHP 3, PHP 4 >= 4.0.0)

Restituisce una riga come array numerato

array **mssql_fetch_row** (int id_risultato) \linebreak

La funzione restituisce un array che corrisponde alla riga estratta, oppure falso se non vi sono più righe.

La funzione **mssql_fetch_row()** estrae una riga di dati dal risultato associato all'identificativo di risultato passato. La riga viene restituita in un array. Ciascuna colonna è memorizzata in un campo dell'array. Il primo ha indice 0.

Esecuzione successive di **mssql_fetch_rows()** restituiscono le righe successive presenti nel risultato, oppure falso se non vi sono più righe.

Vedere anche mssql_fetch_array(), mssql_fetch_object(), mssql_data_seek(), **mssql_fetch_lengths()**, e mssql_result().

mssql_field_length (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Restituisce la lunghezza di un campo

int **mssql_field_length** (int id_risultato [, int offset]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mssql_field_name (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Restituisce il nome di un campo

int **mssql_field_name** (int id_risultato [, int offset]) \linebreak

mssql_field_seek (PHP 3, PHP 4 >= 4.0.0)

Posizionamento sul campo

int **mssql_field_seek** (int id_risultato, int offset_campo) \linebreak

Si posiziona sul campo richiesto. Eseguendo successivamente la funzione mssql_fetch_field() senza indicare alcun campo, quest'ultima restituirà il campo richiesto tramite mssql_fetch_field().

Vedere anche mssql_fetch_field().

mssql_field_type (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Restituisce il tipo di un campo

string **mssql_field_type** (int id_risultato [, int offset]) \linebreak

mssql_free_result (PHP 3, PHP 4 >= 4.0.0)

Libera la memoria di un risultato

int **mssql_free_result** (int id_risultato) \linebreak

E' necessario l'utilizzo della funzione **mssql_free_result()** solo quando si è preoccupati dell'occupazione di memoria durante l'esecuzione dello script. Normalmente tutti i dati verranno rimossi automaticamente dalla memoria al termine dell'esecuzione dello script.E' tuttavia possibile eseguire **mssql_free_result()**, per liberare la memoria occupata dai dati indicati dal parametro *id_risultato*

mssql_get_last_message (PHP 3, PHP 4 >= 4.0.0)

Restituisce l'ultimo messaggio dal server (oltre min_message_severity?)

string **mssql_get_last_message** (void) \linebreak

mssql_guid_string (PHP 4 >= 4.1.0)

Converte il GUID dal formato binario a 16 bit al formato stringa

string **mssql_guid_string** (string binary [, int short_format]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mssql_init (PHP 4 >= 4.1.0)

Inizializza una procedura memorizata locale o remota

int **mssql_init** (string sp_name [, int id_connessione]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mssql_min_error_severity (PHP 3, PHP 4 >= 4.0.0)

Setta il livello minimo di errori critici.

void **mssql_min_error_severity** (int livello_critico) \linebreak

mssql_min_message_severity (PHP 3, PHP 4 >= 4.0.0)

Setta li livello critico minimo di messaggi

void **mssql_min_message_severity** (int livello_critico) \linebreak

mssql_next_result (PHP 4 >= 4.0.5)

Muove il puntatore interno al risultato successivo

bool **mssql_next_result** (int id_risultato) \linebreak

Nel caso in cui si eseguano più di una istruzione SQL al server, oppure si eseguano delle procedure memorizzate (stored procedure) con possibilità di molteplici risultati, il server restituirà un set di diversi risultati. Questa funzione verifica se esistono ulteriori risultati dal server. Se effettivamente esiste un'altro risultato, questa funzione libera la memoria dal risultato corrente e si predispone per la ricezione del risultato successivo. La funzione restituisce `TRUE` se è disponibile un'altro risultato, `FALSE` in caso contrario.

Esempio 1. mssql_next_result() Esempio di utilizzo

```
<?php
    $link = mssql_connect ("localhost", "utente", "password");
    mssql_select_db("MyDB", $link);
    $SQL = "Select * from tabella1 select * from tabella2";
    $rs = mssql_query($SQL, $link);
    do {
        while ($row = mssql_fetch_row($rs)) {
        }
    } while (mssql_next_result($rs));
    mssql_free_result($rs);
    mssql_close ($link);
?>
```

mssql_num_fields (PHP 3, PHP 4 >= 4.0.0)

Restituisce il numero di campi in un risultato

int **mssql_num_fields** (int id_risultato) \linebreak

La funzione **mssql_num_fields()** restituisce il numero di campi presenti in un risultato.

Vedere anche: **mssql_db_query()**, mssql_query(), mssql_fetch_field(), e mssql_num_rows().

mssql_num_rows (PHP 3, PHP 4 >= 4.0.0)

Restituisce il numero di righe

int **mssql_num_rows** (string id_risultato) \linebreak

La funzione **mssql_num_rows()** restituisce il numero di righe presenti in un risultato.

Vedere anche: **mssql_db_query()**, mssql_query(), e mssql_fetch_row().

mssql_pconnect (PHP 3, PHP 4 >= 4.0.0)

Apre una connessione persistente con MS SQL

int **mssql_pconnect** ([string nome_server [, string nome_utente [, string password]]]) \linebreak

La funzione restituisce: o un identificativo di connessione persistente, o FALSE se si verifica un errore.

La funzione **mssql_pconnect()** agisce come mssql_connect() tranne che per due differenze.

Prima differenza, quando si cerca di stabilire la connessione, la funzione per prima cosa cerca di trovare una connessione (persistente) già aperta verso lo stesso server, con i medesimi utenti e password. Se ne viene trovata una, la funzione restituisce l'identificativo di quella connessione, invece di stabilirne una nuova.

Seconda differenza, la connessione con il server SQL non verrà chiusa al termine dello script. Il collegamento resterà aperto per utilizzi futuri (la funzione mssql_close() non chiude i collegamenti aperti da **mssql_pconnect()**).

Per questo motivo questo tipo di collegamento viene definito 'persistente'.

mssql_query (PHP 3, PHP 4 >= 4.0.0)

Invia una query a MS SQL

int **mssql_query** (string testo_query [, int id_connessione]) \linebreak

La funzione restituisce un identificativo di risultato in caso di esecuzione corretta, oppure falso in caso di errore.

La funzione **mssql_query()** invia una query al database attivo sul server attraverso la connessione specificata da id_connessione. Se l'argomento id_connessione non viene fornito, si utilizza l'ultima connessione aperta in ordine di tempo. Se non vi sono connessioni aperte, la funzione tenta di stabilire una connessione, come se fosse utilizzata la funzione mssql_connect(), e utilizza quella.

Vedere anche: **mssql_db_query()**, mssql_select_db(), e mssql_connect().

mssql_result (PHP 3, PHP 4 >= 4.0.0)

Restituisce i dati di un risultato

int **mssql_result** (int id_risultato, int i, mixed campo) \linebreak

La funzione **mssql_result()** restituisce il contenuto di una cella da un risultato di una query a MS SQL. L'argomento campo può essere la posizione di un campo, oppure il suo nome, oppure nome tabella punto nome campo (nome_tabella.nome_campo). Se il nome della colonna ha un sostituto, ('select foo as bar from...'), usare quello anzichè il nome originale.

Quando si lavora con risultati abbastanza grossi, si dovrebbe considerare l'utilizzo di funzioni che restituiscono l'intera riga (indicate di seguito), dato che queste restituiscono il contenuto di molte celle in una chiamata sola. Pertanto sono MOLTO più veloci di **mssql_result()**. Da notare inoltre, che specificando la posizione per l'argomento campo, la funzione è molto più veloce rispetto al caso in cui si indica il nome del campo o della tabella.

Le alternative più veloci raccomandate sono: mssql_fetch_row(), mssql_fetch_array(), e mssql_fetch_object().

mssql_rows_affected (PHP 4 >= 4.0.4)

Restituisce il numero di record coinvolti dalla query

int **mssql_rows_affected** (int id_connessione) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

mssql_select_db (PHP 3, PHP 4 >= 4.0.0)

Seleziona un database MS SQL

int **mssql_select_db** (string Nome_database [, int id_connessione]) \linebreak

Restituisce: vero se l'operazione riesce, falso se si verifica un errore.

La funzione **mssql_select_db()** setta il database attivo sul server attraverso la connessione specificata da id_connessione. Se l'argomento id_connessione non viene fornito, si utilizza l'ultima connessione aperta in ordine di tempo. Se non vi sono connessioni aperte, la funzione tenta di stabilire una connessione, come se fosse utilizzata la funzione mssql_connect(), e utilizza quella.

Ciascuna esecuzione di mssql_query() sarà fatta sul database attivo.

Vedere anche: mssql_connect(), mssql_pconnect(), e mssql_query()

LVIII. Ming functions for Flash

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Introduction

Ming is an open-source (LGPL) library which allows you to create SWF ("Flash") format movies. Ming supports almost all of Flash 4's features, including: shapes, gradients, bitmaps (pngs and jpegs), morphs ("shape tweens"), text, buttons, actions, sprites ("movie clips"), streaming mp3, and color transforms--the only thing that's missing is sound events.

Ming is not an acronym.

Note that all values specifying length, distance, size, etc. are in "twips", twenty units per pixel. That's pretty much arbitrary, though, since the player scales the movie to whatever pixel size is specified in the embed/object tag, or the entire frame if not embedded.

Ming offers a number of advantages over the existing PHP/libswf module. You can use Ming anywhere you can compile the code, whereas libswf is closed-source and only available for a few platforms, Windows not one of them. Ming provides some insulation from the mundane details of the SWF file format, wrapping the movie elements in PHP objects. Also, Ming is still being maintained; if there's a feature that you want to see, just let us know ming@opaque.net (mailto:ming@opaque.net).

Ming was added in PHP 4.0.5.

Installation

To use Ming with PHP, you first need to build and install the Ming library. Source code and installation instructions are available at the Ming home page : http://www.opaque.net/ming/ along with examples, a small tutorial, and the latest news.

Download the ming archive. Unpack the archive. Go in the Ming directory. make. make install.

This will build `libming.so` and install it into `/usr/lib/`, and copy `ming.h` into `/usr/include/`. Edit the `PREFIX=` line in the `Makefile` to change the installation directory.

built into php (unix)

```
mkdir <phpdir>/ext/ming
cp php_ext/* <phpdir>/ext/ming
cd <phpdir>
./buildconf
./configure --with-ming <other config options>
```

Build and install php as usual, Restart web server if necessary

built into php (unix)

download `php_ming.so.gz`. uncompress it and copy it to your php modules directory. (you can find your php module directory by running **php-config --extension-dir**). Now either just add `extension=php_ming.so` to your `php.ini` file, or put `dl('php_ming.so');` at the head of all of your Ming scripts.

How to use Ming

Ming introduces 13 new objects in PHP, all with matching methods and attributes. To use them, you need to know about objects.

- swfmovie().
- swfshape().
- swfdisplayitem().
- swfgradient().
- swfbitmap().
- swffill().
- swfmorph().
- swftext().
- swffont().
- swftextfield().
- swfsprite().
- swfbutton().
- swfaction().

ming_setcubicthreshold (PHP 4 >= 4.0.5)

Set cubic threshold (?)

void **ming_setcubicthreshold** (int threshold) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ming_setscale (PHP 4 >= 4.0.5)

Set scale (?)

void **ming_setscale** (int scale) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

ming_useswfversion (PHP 4 >= 4.2.0)

Use SWF version (?)

void **ming_useswfversion** (int version) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

SWFAction (PHP 4 >= 4.0.5)

Creates a new Action.

new **swfaction** (string script) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfaction() creates a new Action, and compiles the given script into an SWFAction object.

The script syntax is based on the C language, but with a lot taken out- the SWF bytecode machine is just too simpleminded to do a lot of things we might like. For instance, we can't implement function calls without a tremendous amount of hackery because the jump bytecode has a hardcoded offset value. No pushing your calling address to the stack and returning- every function would have to know exactly where to return to.

So what's left? The compiler recognises the following tokens:

- break
- for
- continue
- if
- else
- do
- while

There is no typed data; all values in the SWF action machine are stored as strings. The following functions can be used in expressions:

time()

Returns the number of milliseconds (?) elapsed since the movie started.

random(seed)

Returns a pseudo-random number in the range 0-seed.

length(expr)

Returns the length of the given expression.

int(number)

Returns the given number rounded down to the nearest integer.

concat(expr, expr)

Returns the concatenation of the given expressions.

ord(expr)

Returns the ASCII code for the given character

chr(num)

Returns the character for the given ASCII code

substr(string, location, length)

Returns the substring of length length at location location of the given string string.

Additionally, the following commands may be used:

duplicateClip(clip, name, depth)

Duplicate the named movie clip (aka sprite). The new movie clip has name name and is at depth depth.

removeClip(expr)

Removes the named movie clip.

trace(expr)

Write the given expression to the trace log. Doubtful that the browser plugin does anything with this.

startDrag(target, lock, [left, top, right, bottom])

Start dragging the movie clip target. The lock argument indicates whether to lock the mouse (?)- use 0 (`FALSE`) or 1 (`TRUE`). Optional parameters define a bounding area for the dragging.

stopDrag()

Stop dragging my heart around. And this movie clip, too.

callFrame(expr)

Call the named frame as a function.

getURL(url, target, [method])

Load the given URL into the named target. The target argument corresponds to HTML document targets (such as "_top" or "_blank"). The optional method argument can be POST or GET if you want to submit variables back to the server.

loadMovie(url, target)

Load the given URL into the named target. The target argument can be a frame name (I think), or one of the magical values "_level0" (replaces current movie) or "_level1" (loads new movie on top of current movie).

nextFrame()

Go to the next frame.

prevFrame()

Go to the last (or, rather, previous) frame.

play()

Start playing the movie.

stop()

Stop playing the movie.

toggleQuality()

Toggle between high and low quality.

stopSounds()

Stop playing all sounds.

gotoFrame(num)

Go to frame number num. Frame numbers start at 0.

gotoFrame(name)

Go to the frame named name. Which does a lot of good, since I haven't added frame labels yet.

setTarget(expr)

Sets the context for action. Or so they say- I really have no idea what this does.

And there's one weird extra thing. The expression frameLoaded(num) can be used in if statements and while loops to check if the given frame number has been loaded yet. Well, it's supposed to, anyway, but I've never tested it and I seriously doubt it actually works. You can just use /:framesLoaded instead.

Movie clips (all together now- aka sprites) have properties. You can read all of them (or can you?), you can set some of them, and here they are:

- x
- y
- xScale
- yScale
- currentFrame - (read-only)
- totalFrames - (read-only)
- alpha - transparency level
- visible - 1=on, 0=off (?)
- width - (read-only)
- height - (read-only)
- rotation
- target - (read-only) (???)
- framesLoaded - (read-only)
- name
- dropTarget - (read-only) (???)
- url - (read-only) (???)
- highQuality - 1=high, 0=low (?)
- focusRect - (???)
- soundBufTime - (???)

So, setting a sprite's x position is as simple as `/box.x = 100;`. Why the slash in front of the box, though? That's how flash keeps track of the sprites in the movie, just like a unix filesystem- here it

shows that box is at the top level. If the sprite named box had another sprite named biff inside of it, you'd set its x position with /box/biff.x = 100;. At least, I think so; correct me if I'm wrong here.

This simple example will move the red square across the window.

Esempio 1. swfaction() example

```
<?php
 $s = new SWFShape();
  $f = $s->addFill(0xff, 0, 0);
  $s->setRightFill($f);

  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $i->setDepth(1);
  $p->nextFrame();

  for($n=0; $n<5; ++$n)
  {
    $i->rotate(-15);
    $p->nextFrame();
  }

  $m = new SWFMovie();
  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(6000,4000);

  $i = $m->add($p);
  $i->setDepth(1);
  $i->moveTo(-500,2000);
  $i->setName("box");

  $m->add(new SWFAction("/box.x += 3;"));
  $m->nextFrame();
  $m->add(new SWFAction("gotoFrame(0); play();"));
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example tracks down your mouse on the screen.

Esempio 2. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setRate(36.0);
  $m->setDimension(1200, 800);
  $m->setBackground(0, 0, 0);

  /* mouse tracking sprite - empty, but follows mouse so we can
     get its x and y coordinates */

  $i = $m->add(new SWFSprite());
  $i->setName('mouse');

  $m->add(new SWFAction("
    startDrag('/mouse', 1); /* '1' means lock sprite to the mouse */
  "));

  /* might as well turn off antialiasing, since these are just squares. */

  $m->add(new SWFAction("
    this.quality = 0;
  "));

  /* morphing box */
  $r = new SWFMorph();
  $s = $r->getShape1();

  /* Note this is backwards from normal shapes.  No idea why. */
  $s->setLeftFill($s->addFill(0xff, 0xff, 0xff));
  $s->movePenTo(-40, -40);
  $s->drawLine(80, 0);
  $s->drawLine(0, 80);
  $s->drawLine(-80, 0);
  $s->drawLine(0, -80);

  $s = $r->getShape2();

  $s->setLeftFill($s->addFill(0x00, 0x00, 0x00));
  $s->movePenTo(-1, -1);
  $s->drawLine(2, 0);
  $s->drawLine(0, 2);
  $s->drawLine(-2, 0);
  $s->drawLine(0, -2);

  /* sprite container for morphing box -
     this is just a timeline w/ the box morphing */

  $box = new SWFSprite();
  $box->add(new SWFAction("
    stop();
  "));
  $i = $box->add($r);

  for($n=0; $n<=20; ++$n)
```

```
  {
    $i->setRatio($n/20);
    $box->nextFrame();
  }

  /* this container sprite allows us to use the same action code many times */

  $cell = new SWFSprite();
  $i = $cell->add($box);
  $i->setName('box');

  $cell->add(new SWFAction("

    setTarget('box');

    /* ...x means the x coordinate of the parent, i.e. (..).x */
    dx = (/mouse.x + random(6)-3 - ...x)/5;
    dy = (/mouse.y + random(6)-3 - ...y)/5;
    gotoFrame(int(dx*dx + dy*dy));

  "));

  $cell->nextFrame();
  $cell->add(new SWFAction("

    gotoFrame(0);
    play();

  "));

  $cell->nextFrame();

  /* finally, add a bunch of the cells to the movie */

  for($x=0; $x<12; ++$x)
  {
    for($y=0; $y<8; ++$y)
    {
      $i = $m->add($cell);
      $i->moveTo(100*$x+50, 100*$y+50);
    }
  }

  $m->nextFrame();

  $m->add(new SWFAction("

    gotoFrame(1);
    play();

  "));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

Same as above, but with nice colored balls...

Esempio 3. swfaction() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(11000, 8000);
  $m->setBackground(0x00, 0x00, 0x00);

  $m->add(new SWFAction("

this.quality = 0;
/frames.visible = 0;
startDrag('/mouse', 1);

  "));

  // mouse tracking sprite
  $t = new SWFSprite();
  $i = $m->add($t);
  $i->setName('mouse');

  $g = new SWFGradient();
  $g->addEntry(0, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.1, 0xff, 0xff, 0xff, 0xff);
  $g->addEntry(0.5, 0xff, 0xff, 0xff, 0x5f);
  $g->addEntry(1.0, 0xff, 0xff, 0xff, 0);

  // gradient shape thing
  $s = new SWFShape();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.03);
  $s->setRightFill($f);
  $s->movePenTo(-600, -600);
  $s->drawLine(1200, 0);
  $s->drawLine(0, 1200);
  $s->drawLine(-1200, 0);
  $s->drawLine(0, -1200);

  // need to make this a sprite so we can multColor it
  $p = new SWFSprite();
  $p->add($s);
  $p->nextFrame();

  // put the shape in here, each frame a different color
  $q = new SWFSprite();
  $q->add(new SWFAction("gotoFrame(random(7)+1); stop();"));
  $i = $q->add($p);

  $i->multColor(1.0, 1.0, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 0.5);
  $q->nextFrame();
```

```
  $i->multColor(1.0, 0.75, 0.5);
  $q->nextFrame();
  $i->multColor(1.0, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 1.0, 0.5);
  $q->nextFrame();
  $i->multColor(0.5, 0.5, 1.0);
  $q->nextFrame();
  $i->multColor(1.0, 0.5, 1.0);
  $q->nextFrame();

  // finally, this one contains the action code
  $p = new SWFSprite();
  $i = $p->add($q);
  $i->setName('frames');
  $p->add(new SWFAction("

dx = (/:mousex-/:lastx)/3 + random(10)-5;
dy = (/:mousey-/:lasty)/3;
x = /:mousex;
y = /:mousey;
alpha = 100;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("

this.x = x;
this.y = y;
this.alpha = alpha;
x += dx;
y += dy;
dy += 3;
alpha -= 8;

  "));
  $p->nextFrame();

  $p->add(new SWFAction("prevFrame(); play();"));
  $p->nextFrame();

  $i = $m->add($p);
  $i->setName('frames');
  $m->nextFrame();

  $m->add(new SWFAction("

lastx = mousex;
lasty = mousey;
mousex = /mouse.x;
mousey = /mouse.y;

++num;

if(num == 11)
  num = 1;
```

```
removeClip('char' & num);
duplicateClip(/frames, 'char' & num, num);

  "));

  $m->nextFrame();
  $m->add(new SWFAction("prevFrame(); play();"));

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will handles keyboard actions. (You'll probably have to click in the window to give it focus. And you'll probably have to leave your mouse in the frame, too. If you know how to give buttons focus programatically, feel free to share, won't you?)

Esempio 4. swfaction() example

```
<?php

  /* sprite has one letter per frame */

  $p = new SWFSprite();
  $p->add(new SWFAction("stop();"));

  $chars = "abcdefghijklmnopqrstuvwxyz".
       "ABCDEFGHIJKLMNOPQRSTUVWXYZ".
           "1234567890!@#$%^&/*()_+-=/[]{}|;:,.<>¿~";

  $f = new SWFFont("_sans");

  for($n=0; $nremove($i);
    $t = new SWFTextField();
    $t->setFont($f);
    $t->setHeight(240);
    $t->setBounds(600,240);
    $t->align(SWFTEXTFIELD_ALIGN_CENTER);
    $t->addString($c);
    $i = $p->add($t);
    $p->labelFrame($c);
    $p->nextFrame();
  }

  /* hit region for button - the entire frame */

  $s = new SWFShape();
  $s->setFillStyle0($s->addSolidFill(0, 0, 0, 0));
  $s->drawLine(600, 0);
  $s->drawLine(0, 400);
  $s->drawLine(-600, 0);
  $s->drawLine(0, -400);
```

```
  /* button checks for pressed key, sends sprite to the right frame */

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT);

  for($n=0; $naddAction(new SWFAction("

setTarget('/char');
gotoFrame('$c');

    "), SWFBUTTON_KEYPRESS($c));
  }

  $m = new SWFMovie();
  $m->setDimension(600,400);
  $i = $m->add($p);
  $i->setName('char');
  $i->moveTo(0,80);

  $m->add($b);

  header('Content-type: application/x-shockwave-flash');
  $m->output();

?>
```

SWFBitmap (PHP 4 >= 4.0.5)

Loads Bitmap object

new **swfbitmap** (string filename [, int alphafilename]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbitmap() creates a new SWFBitmap object from the Jpeg or DBL file named *filename*. *alphafilename* indicates a MSK file to be used as an alpha mask for a Jpeg image.

> **Nota:** We can only deal with baseline (frame 0) jpegs, no baseline optimized or progressive scan jpegs!

SWFBitmap has the following methods : **swfbitmap->getwidth()** and **swfbitmap->getheight()**.

You can't import png images directly, though- have to use the png2dbl utility to make a dbl ("define bits lossless") file from the png. The reason for this is that I don't want a dependency on the png library in ming- autoconf should solve this, but that's not set up yet.

Esempio 1. Import PNG files

```
<?php
  $s = new SWFShape();
  $f = $s->addFill(new SWFBitmap("png.dbl"));
  $s->setRightFill($f);

  $s->drawLine(32, 0);
  $s->drawLine(0, 32);
  $s->drawLine(-32, 0);
  $s->drawLine(0, -32);

  $m = new SWFMovie();
  $m->setDimension(32, 32);
  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

And you can put an alpha mask on a jpeg fill.

Esempio 2. swfbitmap() example

```
<?php

  $s = new SWFShape();

  // .msk file generated with "gif2mask" utility
  $f = $s->addFill(new SWFBitmap("alphafill.jpg", "alphafill.msk"));
  $s->setRightFill($f);

  $s->drawLine(640, 0);
  $s->drawLine(0, 480);
  $s->drawLine(-640, 0);
  $s->drawLine(0, -480);

  $c = new SWFShape();
  $c->setRightFill($c->addFill(0x99, 0x99, 0x99));
  $c->drawLine(40, 0);
  $c->drawLine(0, 40);
  $c->drawLine(-40, 0);
  $c->drawLine(0, -40);

  $m = new SWFMovie();
  $m->setDimension(640, 480);
  $m->setBackground(0xcc, 0xcc, 0xcc);
```

```
  // draw checkerboard background
  for($y=0; $y<480; $y+=40)
  {
    for($x=0; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }

    $y+=40;

    for($x=40; $x<640; $x+=80)
    {
      $i = $m->add($c);
      $i->moveTo($x, $y);
    }
  }

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFBitmap->getHeight (unknown)

Returns the bitmap's height.

int **swfbitmap->getheight** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbitmap->getheight() returns the bitmap's height in pixels.

See also **swfbitmap->getwidth()**.

SWFBitmap->getWidth (unknown)

Returns the bitmap's width.

int **swfbitmap->getwidth** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbitmap->getwidth() returns the bitmap's width in pixels.

See also **swfbitmap->getheight()**.

SWFbutton (PHP 4 >= 4.0.5)

Creates a new Button.

new **swfbutton** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbutton() creates a new Button. Roll over it, click it, see it call action code. Swank.

SWFButton has the following methods : **swfbutton->addshape()**, **swfbutton->setup()**, **swfbutton->setover()** **swfbutton->setdown()**, **swfbutton->sethit()** **swfbutton->setaction()** and **swfbutton->addaction()**.

This simple example will show your usual interactions with buttons : rollover, rollon, mouseup, mousedown, noaction.

Esempio 1. swfbutton() example

```
<?php

  $f = new SWFFont("_serif");

  $p = new SWFSprite();

  function label($string)
  {
    global $f;

    $t = new SWFTextField();
    $t->setFont($f);
    $t->addString($string);
    $t->setHeight(200);
    $t->setBounds(3200,200);
    return $t;
  }
  function addLabel($string)
```

```
{
  global $p;

  $i = $p->add(label($string));
  $p->nextFrame();
  $p->remove($i);
}

$p->add(new SWFAction("stop();"));
addLabel("NO ACTION");
addLabel("SWFBUTTON_MOUSEUP");
addLabel("SWFBUTTON_MOUSEDOWN");
addLabel("SWFBUTTON_MOUSEOVER");
addLabel("SWFBUTTON_MOUSEOUT");
addLabel("SWFBUTTON_MOUSEUPOUTSIDE");
addLabel("SWFBUTTON_DRAGOVER");
addLabel("SWFBUTTON_DRAGOUT");

function rect($r, $g, $b)
{
  $s = new SWFShape();
  $s->setRightFill($s->addFill($r, $g, $b));
  $s->drawLine(600,0);
  $s->drawLine(0,600);
  $s->drawLine(-600,0);
  $s->drawLine(0,-600);

  return $s;
}

$b = new SWFButton();
$b->addShape(rect(0xff, 0, 0), SWFBUTTON_UP | SWFBUTTON_HIT);
$b->addShape(rect(0, 0xff, 0), SWFBUTTON_OVER);
$b->addShape(rect(0, 0, 0xff), SWFBUTTON_DOWN);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(1);"),
          SWFBUTTON_MOUSEUP);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(2);"),
      SWFBUTTON_MOUSEDOWN);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(3);"),
      SWFBUTTON_MOUSEOVER);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(4);"),
      SWFBUTTON_MOUSEOUT);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(5);"),
      SWFBUTTON_MOUSEUPOUTSIDE);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(6);"),
      SWFBUTTON_DRAGOVER);

$b->addAction(new SWFAction("setTarget('/label'); gotoFrame(7);"),
      SWFBUTTON_DRAGOUT);

$m = new SWFMovie();
```

```
  $m->setDimension(4000,3000);

  $i = $m->add($p);
  $i->setName("label");
  $i->moveTo(400,1900);

  $i = $m->add($b);
  $i->moveTo(400,900);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

This simple example will enables you to drag draw a big red button on the windows. No drag-and-drop, just moving around.

Esempio 2. swfbutton->addaction() example

```
<?php

  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->drawLine(1000,0);
  $s->drawLine(0,1000);
  $s->drawLine(-1000,0);
  $s->drawLine(0,-1000);

  $b = new SWFButton();
  $b->addShape($s, SWFBUTTON_HIT | SWFBUTTON_UP | SWFBUTTON_DOWN | SWFBUTTON_OVER);

  $b->addAction(new SWFAction("startDrag('/test', 0);"), // '0' means don't lock to mouse
        SWFBUTTON_MOUSEDOWN);

  $b->addAction(new SWFAction("stopDrag();"),
        SWFBUTTON_MOUSEUP | SWFBUTTON_MOUSEUPOUTSIDE);

  $p = new SWFSprite();
  $p->add($b);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->setName('test');
  $i->moveTo(1000,1000);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

swfbutton_keypress (PHP 4 >= 4.0.5)

Returns the action flag for keyPress(char)

int **swfbutton_keypress** (string str) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

SWFbutton->addAction (unknown)

Adds an action

void **swfbutton->addaction** (ressource action, int flags) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbutton->addaction() adds the action *action* to this button for the given conditions. The following *flags* are valid: SWFBUTTON_MOUSEOVER, SWFBUTTON_MOUSEOUT, SWFBUTTON_MOUSEUP, SWFBUTTON_MOUSEUPOUTSIDE, SWFBUTTON_MOUSEDOWN, SWFBUTTON_DRAGOUT and SWFBUTTON_DRAGOVER.

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->addShape (unknown)

Adds a shape to a button

void **swfbutton->addshape** (ressource shape, int flags) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbutton->addshape() adds the shape *shape* to this button. The following *flags*' values are valid: SWFBUTTON_UP, SWFBUTTON_OVER, SWFBUTTON_DOWN or SWFBUTTON_HIT.

SWFBUTTON_HIT isn't ever displayed, it defines the hit region for the button. That is, everywhere the hit shape would be drawn is considered a "touchable" part of the button.

SWFbutton->setAction (unknown)

Sets the action

void **swfbutton->setaction** (ressource action) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbutton->setaction() sets the action to be performed when the button is clicked. Alias for addAction(shape, SWFBUTTON_MOUSEUP). *action* is a swfaction().

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setdown (unknown)

Alias for addShape(shape, SWFBUTTON_DOWN))

void **swfbutton->setdown** (ressource shape) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbutton->setdown() alias for addShape(shape, SWFBUTTON_DOWN).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setHit (unknown)

Alias for addShape(shape, SWFBUTTON_HIT)

void **swfbutton->sethit** (ressource shape) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbutton->sethit() alias for addShape(shape, SWFBUTTON_HIT).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setOver (unknown)

Alias for addShape(shape, SWFBUTTON_OVER)

void **swfbutton->setover** (ressource shape) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbutton->setover() alias for addShape(shape, SWFBUTTON_OVER).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFbutton->setUp (unknown)

Alias for addShape(shape, SWFBUTTON_UP)

void **swfbutton->setup** (ressource shape) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfbutton->setup() alias for addShape(shape, SWFBUTTON_UP).

See also **swfbutton->addshape()** and **SWFAction()**.

SWFDisplayItem (unknown)

Creates a new displayitem object.

new **swfdisplayitem** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem() creates a new swfdisplayitem object.

Here's where all the animation takes place. After you define a shape, a text object, a sprite, or a button, you add it to the movie, then use the returned handle to move, rotate, scale, or skew the thing.

SWFDisplayItem has the following methods : **swfdisplayitem->move()**, **swfdisplayitem->moveto()**, **swfdisplayitem->scaleto()**, **swfdisplayitem->scale()**, **swfdisplayitem->rotate()**, **swfdisplayitem->rotateto()**, **swfdisplayitem->skewxto()**, **swfdisplayitem->skewx()**, **swfdisplayitem->skewyto() swfdisplayitem->skewyto()**, **swfdisplayitem->setdepth() swfdisplayitem->remove()**, **swfdisplayitem->setname() swfdisplayitem->setratio()**, **swfdisplayitem->addcolor()** and **swfdisplayitem->multcolor()**.

SWFDisplayItem->addColor (unknown)

Adds the given color to this item's color transform.

void **swfdisplayitem->addcolor** ([int red [, int green [, int blue [, int a]]]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->addcolor() adds the color to this item's color transform. The color is given in its RGB form.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->move (unknown)

Moves object in relative coordinates.

void **swfdisplayitem->move** (int dx, int dy) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->move() moves the current object by (*dx*,*dy*) from its current position.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->moveto()**.

SWFDisplayItem->moveTo (unknown)

Moves object in global coordinates.

void **swfdisplayitem->moveto** (int x, int y) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->moveto() moves the current object to (*x*,*y*) in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->move()**.

SWFDisplayItem->multColor (unknown)

Multiplies the item's color transform.

void **swfdisplayitem->multcolor** ([int red [, int green [, int blue [, int a]]]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->multcolor() multiplies the item's color transform by the given values.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This simple example will modify your picture's atmospher to Halloween (use a landscape or bright picture).

Esempio 1. swfdisplayitem->multcolor() example

```
<?php

  $b = new SWFBitmap("backyard.jpg");
  // note use your own picture :-)
  $s = new SWFShape();
  $s->setRightFill($s->addFill($b));
  $s->drawLine($b->getWidth(), 0);
  $s->drawLine(0, $b->getHeight());
  $s->drawLine(-$b->getWidth(), 0);
  $s->drawLine(0, -$b->getHeight());

  $m = new SWFMovie();
  $m->setDimension($b->getWidth(), $b->getHeight());

  $i = $m->add($s);

  for($n=0; $n<=20; ++$n)
  {
    $i->multColor(1.0-$n/10, 1.0, 1.0);
    $i->addColor(0xff*$n/20, 0, 0);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->remove (unknown)

Removes the object from the movie

void **swfdisplayitem->remove** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->remove() removes this object from the movie's display list.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfmovie->add()**.

SWFDisplayItem->Rotate (unknown)

Rotates in relative coordinates.

void **swfdisplayitem->rotate** (float ddegrees) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->rotate() rotates the current object by *ddegrees* degrees from its current rotation.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->rotateto()**.

SWFDisplayItem->rotateTo (unknown)

Rotates the object in global coordinates.

void **swfdisplayitem->rotateto** (float degrees) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->rotateto() set the current object rotation to *degrees* degrees in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This example bring three rotating string from the background to the foreground. Pretty nice.

Esempio 1. swfdisplayitem->rotateto() example

```
<?php
  $thetext =  "ming!";

  $f = new SWFFont("Bauhaus 93.fdb");
```

```
$m = new SWFMovie();
$m->setRate(24.0);
$m->setDimension(2400, 1600);
$m->setBackground(0xff, 0xff, 0xff);

// functions with huge numbers of arbitrary
// arguments are always a good idea!  Really!

function text($r, $g, $b, $a, $rot, $x, $y, $scale, $string)
{
  global $f, $m;

  $t = new SWFText();
  $t->setFont($f);
  $t->setColor($r, $g, $b, $a);
  $t->setHeight(960);
  $t->moveTo(-($f->getWidth($string))/2, $f->getAscent()/2);
  $t->addString($string);

  // we can add properties just like a normal php var,
  // as long as the names aren't already used.
  // e.g., we can't set $i->scale, because that's a function

  $i = $m->add($t);
  $i->x = $x;
  $i->y = $y;
  $i->rot = $rot;
  $i->s = $scale;
  $i->rotateTo($rot);
  $i->scale($scale, $scale);

  // but the changes are local to the function, so we have to
  // return the changed object.  kinda weird..

  return $i;
}

function step($i)
{
  $oldrot = $i->rot;
  $i->rot = 19*$i->rot/20;
  $i->x = (19*$i->x + 1200)/20;
  $i->y = (19*$i->y + 800)/20;
  $i->s = (19*$i->s + 1.0)/20;

  $i->rotateTo($i->rot);
  $i->scaleTo($i->s, $i->s);
  $i->moveTo($i->x, $i->y);

  return $i;
}

// see?  it sure paid off in legibility:

$i1 = text(0xff, 0x33, 0x33, 0xff, 900, 1200, 800, 0.03, $thetext);
$i2 = text(0x00, 0x33, 0xff, 0x7f, -560, 1200, 800, 0.04, $thetext);
$i3 = text(0xff, 0xff, 0xff, 0x9f, 180, 1200, 800, 0.001, $thetext);
```

```
  for($i=1; $i<=100; ++$i)
  {
    $i1 = step($i1);
    $i2 = step($i2);
    $i3 = step($i3);

    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

See also **swfdisplayitem->rotate()**.

SWFDisplayItem->scale (unknown)

Scales the object in relative coordinates.

void **swfdisplayitem->scale** (int dx, int dy) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->scale() scales the current object by (*dx*,*dy*) from its current size.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scaleto()**.

SWFDisplayItem->scaleTo (unknown)

Scales the object in global coordinates.

void **swfdisplayitem->scaleto** (int x, int y) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->scaleto() scales the current object to (x,y) in global coordinates.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->scale()**.

SWFDisplayItem->setDepth (unknown)

Sets z-order

void **swfdisplayitem->setdepth** (float depth) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->rotate() sets the object's z-order to *depth*. Depth defaults to the order in which instances are created (by add'ing a shape/text to a movie)- newer ones are on top of older ones. If two objects are given the same depth, only the later-defined one can be moved.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setName (unknown)

Sets the object's name

void **swfdisplayitem->setname** (string name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->setname() sets the object's name to *name*, for targetting with action script. Only useful on sprites.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

SWFDisplayItem->setRatio (unknown)

Sets the object's ratio.

void **swfdisplayitem->setratio** (float ratio) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->setratio() sets the object's ratio to `ratio`. Obviously only useful for morphs.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

This simple example will morph nicely three concentric circles.

Esempio 1. swfdisplayitem->setname() example

```
<?php

  $p = new SWFMorph();

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0xff, 0xff);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0xff, 0xff, 0xff);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0xff, 0xff, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape1();
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(0.16, 0xff, 0, 0);
  $g->addEntry(0.32, 0, 0, 0);
  $g->addEntry(0.48, 0, 0xff, 0);
  $g->addEntry(0.64, 0, 0, 0);
  $g->addEntry(0.80, 0, 0, 0xff);
  $g->addEntry(1.00, 0, 0, 0);

  $s = $p->getShape2();
```

```
  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.05);
  $f->skewXTo(1.0);
  $s->setLeftFill($f);
  $s->movePenTo(-160, -120);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m = new SWFMovie();
  $m->setDimension(320, 240);
  $i = $m->add($p);
  $i->moveTo(160, 120);

  for($n=0; $n<=1.001; $n+=0.01)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFDisplayItem->skewX (unknown)

Sets the X-skew.

void **swfdisplayitem->skewx** (float ddegrees) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->skewx() adds `ddegrees` to current x-skew.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewXTo (unknown)

Sets the X-skew.

void **swfdisplayitem->skewxto** (float degrees) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->skewxto() sets the x-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more forward, less is more backward.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewx()**, **swfdisplayitem->skewy()** and **swfdisplayitem->skewyto()**.

SWFDisplayItem->skewY (unknown)

Sets the Y-skew.

void **swfdisplayitem->skewy** (float ddegrees) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->skewy() adds `ddegrees` to current y-skew.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewyto()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFDisplayItem->skewYTo (unknown)

Sets the Y-skew.

void **swfdisplayitem->skewyto** (float degrees) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfdisplayitem->skewyto() sets the y-skew to `degrees`. For `degrees` is 1.0, it means a 45-degree forward slant. More is more upward, less is more downward.

The object may be a swfshape(), a swfbutton(), a swftext() or a swfsprite() object. It must have been added using the **swfmovie->add()**.

See also **swfdisplayitem->skewy()**, **swfdisplayitem->skewx()** and **swfdisplayitem->skewxto()**.

SWFFill (PHP 4 >= 4.0.5)

Loads SWFFill object

The **swffill()** object allows you to transform (scale, skew, rotate) bitmap and gradient fills. **swffill()** objects are created by the **swfshape->addfill()** methods.

SWFFill has the following methods : **swffill->moveto()** and **swffill->scaleto()**, **swffill->rotateto()**, **swffill->skewxto()** and **swffill->skewyto()**.

SWFFill->moveTo (unknown)

Moves fill origin

void **swffill->moveto** (int x, int y) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swffill->moveto() moves fill's origin to (`x`,`y`) in global coordinates.

SWFFill->rotateTo (unknown)

Sets fill's rotation

void **swffill->rotateto** (float degrees) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swffill->rotateto() sets fill's rotation to *degrees* degrees.

SWFFill->scaleTo (unknown)

Sets fill's scale

void **swffill->scaleto** (int x, int y) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swffill->scaleto() sets fill's scale to *x* in the x-direction, *y* in the y-direction.

SWFFill->skewXTo (unknown)

Sets fill x-skew

void **swffill->skewxto** (float x) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swffill->skewxto() sets fill x-skew to *x*. For *x* is 1.0, it is a is a 45-degree forward slant. More is more forward, less is more backward.

SWFFill->skewYTo (unknown)

Sets fill y-skew

void **swffill->skewyto** (float y) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swffill->skewyto() sets fill y-skew to *y*. For *y* is 1.0, it is a is a 45-degree upward slant. More is more upward, less is more downward.

SWFFont (PHP 4 >= 4.0.5)

Loads a font definition

new **swffont** (string filename) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

If *filename* is the name of an FDB file (i.e., it ends in ".fdb"), load the font definition found in said file. Otherwise, create a browser-defined font reference.

FDB ("font definition block") is a very simple wrapper for the SWF DefineFont2 block which contains a full description of a font. One may create FDB files from SWT Generator template files with the included makefdb utility- look in the util directory off the main ming distribution directory.

Browser-defined fonts don't contain any information about the font other than its name. It is assumed that the font definition will be provided by the movie player. The fonts _serif, _sans, and _typewriter should always be available. For example:

```
<?php
$f = newSWFFont("_sans");
?>
```

will give you the standard sans-serif font, probably the same as what you'd get with `<font name="sans-serif">` in HTML.

swffont() returns a reference to the font definition, for use in the **SWFText->setFont()** and the **SWFTextField->setFont()** methods.

SWFFont has the following methods : **swffont->getwidth()**.

swffont->getwidth (unknown)

Returns the string's width

int **swffont->getwidth** (string string) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swffont->getwidth() returns the string *`string`*'s width, using font's default scaling. You'll probably want to use the **SWFText()** version of this method which uses the text object's scale.

SWFGradient (PHP 4 >= 4.0.5)

Creates a gradient object

new **swfgradient** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfgradient() creates a new SWFGradient object.

After you've added the entries to your gradient, you can use the gradient in a shape fill with the **swfshape->addfill()** method.

SWFGradient has the following methods : **swfgradient->addentry()**.

This simple example will draw a big black-to-white gradient as background, and a redish disc in its center.

Esempio 1. swfgradient() example

```
<?php

  $m = new SWFMovie();
  $m->setDimension(320, 240);

  $s = new SWFShape();

  // first gradient- black to white
  $g = new SWFGradient();
  $g->addEntry(0.0, 0, 0, 0);
  $g->addEntry(1.0, 0xff, 0xff, 0xff);
```

```
  $f = $s->addFill($g, SWFFILL_LINEAR_GRADIENT);
  $f->scaleTo(0.01);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  $s = new SWFShape();

  // second gradient- radial gradient from red to transparent
  $g = new SWFGradient();
  $g->addEntry(0.0, 0xff, 0, 0, 0xff);
  $g->addEntry(1.0, 0xff, 0, 0, 0);

  $f = $s->addFill($g, SWFFILL_RADIAL_GRADIENT);
  $f->scaleTo(0.005);
  $f->moveTo(160, 120);
  $s->setRightFill($f);
  $s->drawLine(320, 0);
  $s->drawLine(0, 240);
  $s->drawLine(-320, 0);
  $s->drawLine(0, -240);

  $m->add($s);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFGradient->addEntry (unknown)

Adds an entry to the gradient list.

void **swfgradient->addentry** (float ratio, int red, int green, int blue [, int a]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfgradient->addentry() adds an entry to the gradient list. `ratio` is a number between 0 and 1 indicating where in the gradient this color appears. Thou shalt add entries in order of increasing ratio.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

SWFMorph (PHP 4 >= 4.0.5)

Creates a new SWFMorph object.

new **swfmorph** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmorph() creates a new SWFMorph object.

Also called a "shape tween". This thing lets you make those tacky twisting things that make your computer choke. Oh, joy!

The methods here are sort of weird. It would make more sense to just have newSWFMorph(shape1, shape2);, but as things are now, shape2 needs to know that it's the second part of a morph. (This, because it starts writing its output as soon as it gets drawing commands- if it kept its own description of its shapes and wrote on completion this and some other things would be much easier.)

SWFMorph has the following methods : **swfmorph->getshape1()** and **swfmorph->getshape1()**.

This simple example will morph a big red square into a smaller blue black-bordered square.

Esempio 1. swfmorph() example

```
<?php
  $p = new SWFMorph();

  $s = $p->getShape1();
  $s->setLine(0,0,0,0);

  /* Note that this is backwards from normal shapes (left instead of right).
     I have no idea why, but this seems to work.. */

  $s->setLeftFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-1000,-1000);
  $s->drawLine(2000,0);
  $s->drawLine(0,2000);
  $s->drawLine(-2000,0);
  $s->drawLine(0,-2000);

  $s = $p->getShape2();
  $s->setLine(60,0,0,0);
  $s->setLeftFill($s->addFill(0, 0, 0xff));
  $s->movePenTo(0,-1000);
  $s->drawLine(1000,1000);
  $s->drawLine(-1000,1000);
  $s->drawLine(-1000,-1000);
```

```
  $s->drawLine(1000,-1000);

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setBackground(0xff, 0xff, 0xff);

  $i = $m->add($p);
  $i->moveTo(1500,1000);

  for($r=0.0; $r<=1.0; $r+=0.1)
  {
    $i->setRatio($r);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFMorph->getshape1 (unknown)

Gets a handle to the starting shape

mixed **swfmorph->getshape1** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmorph->getshape1() gets a handle to the morph's starting shape. **swfmorph->getshape1()** returns an swfshape() object.

SWFMorph->getshape2 (unknown)

Gets a handle to the ending shape

mixed **swfmorph->getshape2** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmorph->getshape2() gets a handle to the morph's ending shape. **swfmorph->getshape2()** returns an swfshape() object.

SWFMovie (PHP 4 >= 4.0.5)

Creates a new movie object, representing an SWF version 4 movie.

new **swfmovie** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie() creates a new movie object, representing an SWF version 4 movie.

SWFMovie has the following methods : **swfmovie->output()**,**swfmovie->save()**, **swfmovie->add()**, **swfmovie->remove()**, **swfmovie->nextframe()**, **swfmovie->setbackground()**, **swfmovie->setrate()**, **swfmovie->setdimension()**, **swfmovie->setframes()** and **swfmovie->streammp3()**.

See examples in : **swfdisplayitem->rotateto()**, **swfshape->setline()**, **swfshape->addfill()**... Any example will use this object.

SWFMovie->add (unknown)

Adds any type of data to a movie.

void **swfmovie->add** (ressource instance) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->add() adds `instance` to the current movie. `instance` is any type of data : Shapes, text, fonts, etc. must all be add'ed to the movie to make this work.

For displayable types (shape, text, button, sprite), this returns an **SWFDisplayItem()**, a handle to the object in a display list. Thus, you can add the same shape to a movie multiple times and get separate handles back for each separate instance.

See also all other objects (adding this later), and **swfmovie->remove()**

See examples in : **swfdisplayitem->rotateto()** and **swfshape->addfill()**.

SWFMovie->nextframe (unknown)

Moves to the next frame of the animation.

void **swfmovie->nextframe** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->setframes() moves to the next frame of the animation.

SWFMovie->output (unknown)

Dumps your lovingly prepared movie out.

void **swfmovie->output** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->output() dumps your lovingly prepared movie out. In PHP, preceding this with the command

```
<?php
header('Content-type: application/x-shockwave-flash');
?>
```

convinces the browser to display this as a flash movie.

See also **swfmovie->save()**.

See examples in : **swfmovie->streammp3()**, **swfdisplayitem->rotateto()**, swfaction()... Any example will use this method.

SWFMovie->remove (unknown)

Removes the object instance from the display list.

void **swfmovie->remove** (resource instance) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->remove() removes the object instance *instance* from the display list.

See also **swfmovie->add()**.

SWFMovie->save (unknown)

Saves your movie in a file.

void **swfmovie->save** (string filename) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->save() saves your movie to the file named *filename*.

See also **output()**.

SWFMovie->setbackground (unknown)

Sets the background color.

void **swfmovie->setbackground** (int red, int green, int blue) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->setbackground() sets the background color. Why is there no rgba version? Think about it. (Actually, that's not such a dumb question after all- you might want to let the html background

show through. There's a way to do that, but it only works on IE4. Search the http://www.macromedia.com/ site for details.)

SWFMovie->setdimension (unknown)

Sets the movie's width and height.

void **swfmovie->setdimension** (int width, int height) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->setdimension() sets the movie's width to *`width`* and height to *`height`*.

SWFMovie->setframes (unknown)

Sets the total number of frames in the animation.

void **swfmovie->setframes** (string numberofframes) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->setframes() sets the total number of frames in the animation to *`numberofframes`*.

SWFMovie->setrate (unknown)

Sets the animation's frame rate.

void **swfmovie->setrate** (int rate) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->setrate() sets the frame rate to *rate*, in frame per seconds. Animation will slow down if the player can't render frames fast enough- unless there's a streaming sound, in which case display frames are sacrificed to keep sound from skipping.

SWFMovie->streammp3 (unknown)

Streams a MP3 file.

void **swfmovie->streammp3** (string mp3FileName) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfmovie->streammp3() streams the mp3 file *mp3FileName*. Not very robust in dealing with oddities (can skip over an initial ID3 tag, but that's about it). Like **SWFShape->addJpegFill()**, this isn't a stable function- we'll probably need to make a separate SWFSound object to contain sound types.

Note that the movie isn't smart enough to put enough frames in to contain the entire mp3 stream- you'll have to add (length of song * frames per second) frames to get the entire stream in.

Yes, now you can use ming to put that rock and roll devil worship music into your SWF files. Just don't tell the RIAA.

Esempio 1. swfmovie->streammp3() example

```
<?php
  $m = new SWFMovie();
  $m->setRate(12.0);
  $m->streamMp3("distortobass.mp3");
  // use your own MP3

  // 11.85 seconds at 12.0 fps = 142 frames
  $m->setFrames(142);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape (PHP 4 >= 4.0.5)

Creates a new shape object.

new **swfshape** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape() creates a new shape object.

SWFShape has the following methods : **swfshape->setline()**, **swfshape->addfill()**, **swfshape->setleftfill()**, **swfshape->setrightfill()**, **swfshape->movepento()**, **swfshape->movepen()**, **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->drawcurveto()** and **swfshape->drawcurve()**.

This simple example will draw a big red elliptic quadrant.

Esempio 1. swfshape() example

```
<?php
  $s = new SWFShape();
  $s->setLine(40, 0x7f, 0, 0);
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(200, 200);
  $s->drawLineTo(6200, 200);
  $s->drawLineTo(6200, 4600);
  $s->drawCurveTo(200, 4600, 200, 200);

  $m = new SWFMovie();
  $m->setDimension(6400, 4800);
  $m->setRate(12.0);
  $m->add($s);
  $m->nextFrame();

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->addFill (unknown)

Adds a solid fill to the shape.

void **swfshape->addfill** (int red, int green, int blue [, int a]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

void **swfshape->addfill** (SWFbitmap bitmap [, int flags]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

void **swfshape->addfill** (SWFGradient gradient [, int flags]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape->addfill() adds a solid fill to the shape's list of fill styles. **swfshape->addfill()** accepts three different types of arguments.

red, *green*, *blue* is a color (RGB mode). Last parameter *a* is optional.

The *bitmap* argument is an swfbitmap() object. The *flags* argument can be one of the following values : SWFFILL_CLIPPED_BITMAP or SWFFILL_TILED_BITMAP. Default is SWFFILL_TILED_BITMAP. I think.

The *gradient* argument is an swfgradient() object. The flags argument can be one of the following values : SWFFILL_RADIAL_GRADIENT or SWFFILL_LINEAR_GRADIENT. Default is SWFFILL_LINEAR_GRADIENT. I'm sure about this one. Really.

swfshape->addfill() returns an swffill() object for use with the **swfshape->setleftfill()** and **swfshape->setrightfill()** functions described below.

See also **swfshape->setleftfill()** and **swfshape->setrightfill()**.

This simple example will draw a frame on a bitmap. Ah, here's another buglet in the flash player- it doesn't seem to care about the second shape's bitmap's transformation in a morph. According to spec, the bitmap should stretch along with the shape in this example..

Esempio 1. swfshape->addfill() example

```
<?php

  $p = new SWFMorph();

  $b = new SWFBitmap("alphafill.jpg");
  // use your own bitmap
```

```
  $width = $b->getWidth();
  $height = $b->getHeight();

  $s = $p->getShape1();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);
  $f->moveTo(-$width/2, -$height/4);
  $f->scaleTo(1.0, 0.5);
  $s->setLeftFill($f);
  $s->movePenTo(-$width/2, -$height/4);
  $s->drawLine($width, 0);
  $s->drawLine(0, $height/2);
  $s->drawLine(-$width, 0);
  $s->drawLine(0, -$height/2);

  $s = $p->getShape2();
  $f = $s->addFill($b, SWFFILL_TILED_BITMAP);

  // these two have no effect!
  $f->moveTo(-$width/4, -$height/2);
  $f->scaleTo(0.5, 1.0);

  $s->setLeftFill($f);
  $s->movePenTo(-$width/4, -$height/2);
  $s->drawLine($width/2, 0);
  $s->drawLine(0, $height);
  $s->drawLine(-$width/2, 0);
  $s->drawLine(0, -$height);

  $m = new SWFMovie();
  $m->setDimension($width, $height);
  $i = $m->add($p);
  $i->moveTo($width/2, $height/2);

  for($n=0; $n<1.001; $n+=0.03)
  {
    $i->setRatio($n);
    $m->nextFrame();
  }

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->drawCurve (unknown)

Draws a curve (relative).

void **swfshape->drawcurve** (int controldx, int controldy, int anchordx, int anchordy) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape->drawcurve() draws a quadratic curve (using the current line style,set by **swfshape->setline()**) from the current pen position to the relative position (*anchorx*,*anchory*) using relative control point (*controlx*,*controly*). That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawCurveTo (unknown)

Draws a curve.

void **swfshape->drawcurveto** (int controlx, int controly, int anchorx, int anchory) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape->drawcurveto() draws a quadratic curve (using the current line style, set by **swfshape->setline()**) from the current pen position to (*anchorx*,*anchory*) using (*controlx*,*controly*) as a control point. That is, head towards the control point, then smoothly turn to the anchor point.

See also **swfshape->drawlineto()**, **swfshape->drawline()**, **swfshape->movepento()** and **swfshape->movepen()**.

SWFShape->drawLine (unknown)

Draws a line (relative).

void **swfshape->drawline** (int dx, int dy) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape->drawline() draws a line (using the current line style set by **swfshape->setline()**) from the current pen position to displacement (*dx*,*dy*).

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawlineto()**.

SWFShape->drawLineTo (unknown)

Draws a line.

void **swfshape->drawlineto** (int x, int y) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape->setrightfill() draws a line (using the current line style, set by **swfshape->setline()**) from the current pen position to point (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->movepen()** and **swfshape->drawline()**.

SWFShape->movePen (unknown)

Moves the shape's pen (relative).

void **swfshape->movepen** (int dx, int dy) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape->setrightfill() move the shape's pen from coordinates (current x,current y) to (current x + *dx*, current y + *dy*) in the shape's coordinate space.

See also **swfshape->movepento()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->movePenTo (unknown)

Moves the shape's pen.

void **swfshape->movepento** (int x, int y) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape->setrightfill() move the shape's pen to (*x*,*y*) in the shape's coordinate space.

See also **swfshape->movepen()**, **swfshape->drawcurveto()**, **swfshape->drawlineto()** and **swfshape->drawline()**.

SWFShape->setLeftFill (unknown)

Sets left rasterizing color.

void **swfshape->setleftfill** (swfgradient fill) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

void **swfshape->setleftfill** (int red, int green, int blue [, int a]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

What this nonsense is about is, every edge segment borders at most two fills. When rasterizing the object, it's pretty handy to know what those fills are ahead of time, so the swf format requires these to be specified.

swfshape->setleftfill() sets the fill on the left side of the edge- that is, on the interior if you're defining the outline of the shape in a counter-clockwise fashion. The fill object is an SWFFill object returned from one of the addFill functions above.

This seems to be reversed when you're defining a shape in a morph, though. If your browser crashes, just try setting the fill on the other side.

Shortcut for `swfshape->setleftfill($s->addfill($r, $g, $b [, $a]));`.

See also **swfshape->setrightfill()**.

SWFShape->setLine (unknown)

Sets the shape's line style.

void **swfshape->setline** (int width [, int red [, int green [, int blue [, int a]]]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfshape->setline() sets the shape's line style. `width` is the line's width. If `width` is 0, the line's style is removed (then, all other arguments are ignored). If `width` > 0, then line's color is set to `red`, `green`, `blue`. Last parameter `a` is optional.

swfshape->setline() accepts 1, 4 or 5 arguments (not 3 or 2).

You must declare all line styles before you use them (see example).

This simple example will draw a big "!#%*@", in funny colors and gracious style.

Esempio 1. swfshape->setline() example

```
<?php
  $s = new SWFShape();
  $f1 = $s->addFill(0xff, 0, 0);
  $f2 = $s->addFill(0xff, 0x7f, 0);
  $f3 = $s->addFill(0xff, 0xff, 0);
  $f4 = $s->addFill(0, 0xff, 0);
  $f5 = $s->addFill(0, 0, 0xff);

  // bug: have to declare all line styles before you use them
  $s->setLine(40, 0x7f, 0, 0);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->setLine(40, 0, 0x7f, 0);
  $s->setLine(40, 0, 0, 0x7f);

  $f = new SWFFont('Techno.fdb');

  $s->setRightFill($f1);
  $s->setLine(40, 0x7f, 0, 0);
  $s->drawGlyph($f, '!');
  $s->movePen($f->getWidth('!'), 0);

  $s->setRightFill($f2);
  $s->setLine(40, 0x7f, 0x3f, 0);
  $s->drawGlyph($f, '#');
  $s->movePen($f->getWidth('#'), 0);

  $s->setRightFill($f3);
  $s->setLine(40, 0x7f, 0x7f, 0);
  $s->drawGlyph($f, '%');
  $s->movePen($f->getWidth('%'), 0);
```

```
  $s->setRightFill($f4);
  $s->setLine(40, 0, 0x7f, 0);
  $s->drawGlyph($f, '*');
  $s->movePen($f->getWidth('*'), 0);

  $s->setRightFill($f5);
  $s->setLine(40, 0, 0, 0x7f);
  $s->drawGlyph($f, '@');

  $m = new SWFMovie();
  $m->setDimension(3000,2000);
  $m->setRate(12.0);
  $i = $m->add($s);
  $i->moveTo(1500-$f->getWidth("!#%*@")/2, 1000+$f->getAscent()/2);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFShape->setRightFill (unknown)

Sets right rasterizing color.

void **swfshape->setrightfill** (swfgradient fill) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

void **swfshape->setrightfill** (int red, int green, int blue [, int a]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

See also **swfshape->setleftfill()**.

Shortcut for `swfshape->setrightfill($s->addfill($r, $g, $b [, $a]));`.

SWFSprite (PHP 4 >= 4.0.5)

Creates a movie clip (a sprite)

new **swfsprite** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfsprite() are also known as a "movie clip", this allows one to create objects which are animated in their own timelines. Hence, the sprite has most of the same methods as the movie.

swfsprite() has the following methods : **swfsprite->add()**, **swfsprite->remove()**, **swfsprite->nextframe()** and **swfsprite->setframes()**.

This simple example will spin gracefully a big red square.

Esempio 1. swfsprite() example

```
<?php
  $s = new SWFShape();
  $s->setRightFill($s->addFill(0xff, 0, 0));
  $s->movePenTo(-500,-500);
  $s->drawLineTo(500,-500);
  $s->drawLineTo(500,500);
  $s->drawLineTo(-500,500);
  $s->drawLineTo(-500,-500);

  $p = new SWFSprite();
  $i = $p->add($s);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();
  $i->rotate(15);
  $p->nextFrame();

  $m = new SWFMovie();
  $i = $m->add($p);
  $i->moveTo(1500,1000);
  $i->setName("blah");

  $m->setBackground(0xff, 0xff, 0xff);
  $m->setDimension(3000,2000);

  header('Content-type: application/x-shockwave-flash');
```

```
  $m->output();
?>
```

SWFSprite->add (unknown)

Adds an object to a sprite

void **swfsprite->add** (resource object) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfsprite->add() adds a swfshape(), a swfbutton(), a swftext(), a swfaction() or a swfsprite() object.

For displayable types (swfshape(), swfbutton(), swftext(), swfaction() or swfsprite()), this returns a handle to the object in a display list.

SWFSprite->nextframe (unknown)

Moves to the next frame of the animation.

void **swfsprite->nextframe** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfsprite->setframes() moves to the next frame of the animation.

SWFSprite->remove (unknown)

Removes an object to a sprite

void **swfsprite->remove** (ressource object) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfsprite->remove() remove a swfshape(), a swfbutton(), a swftext(), a swfaction() or a swfsprite() object from the sprite.

SWFSprite->setframes (unknown)

Sets the total number of frames in the animation.

void **swfsprite->setframes** (int numberofframes) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swfsprite->setframes() sets the total number of frames in the animation to *`numberofframes`*.

SWFText (PHP 4 >= 4.0.5)

Creates a new SWFText object.

new **swftext** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftext() creates a new SWFText object, fresh for manipulating.

SWFText has the following methods : **swftext->setfont()**, **swftext->setheight()**, **swftext->setspacing()**, **swftext->setcolor()**, **swftext->moveto()**, **swftext->addstring()** and **swftext->getwidth()**.

This simple example will draw a big yellow "PHP generates Flash with Ming" text, on white background.

Esempio 1. swftext() example

```
<?php
  $f = new SWFFont("Techno.fdb");
  $t = new SWFText();
  $t->setFont($f);
  $t->moveTo(200, 2400);
  $t->setColor(0xff, 0xff, 0);
  $t->setHeight(1200);
  $t->addString("PHP generates Flash with Ming!!");

  $m = new SWFMovie();
  $m->setDimension(5400, 3600);

  $m->add($t);

  header('Content-type: application/x-shockwave-flash');
  $m->output();
?>
```

SWFText->addString (unknown)

Draws a string

void **swftext->addstring** (string string) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftext->addstring() draws the string *string* at the current pen (cursor) location. Pen is at the baseline of the text; i.e., ascending text is in the -y direction.

SWFText->getWidth (unknown)

Computes string's width

void **swftext->addstring** (string string) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

swftext->addstring() returns the rendered width of the `string` string at the text object’s current font, scale, and spacing settings.

SWFText->moveTo (unknown)

Moves the pen

void **swftext->moveto** (int x, int y) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

swftext->moveto() moves the pen (or cursor, if that makes more sense) to (*x*,*y*) in text object’s coordinate space. If either is zero, though, value in that dimension stays the same. Annoying, should be fixed.

SWFText->setColor (unknown)

Sets the current font color

void **swftext->setcolor** (int red, int green, int blue [, int a]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

swftext->setspacing() changes the current text color. Default is black. I think. Color is represented using the RGB system.

SWFText->setFont (unknown)

Sets the current font

void **swftext->setfont** (string font) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftext->setfont() sets the current font to `font`.

SWFText->setHeight (unknown)

Sets the current font height

void **swftext->setheight** (int height) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftext->setheight() sets the current font height to `height`. Default is 240.

SWFText->setSpacing (unknown)

Sets the current font spacing

void **swftext->setspacing** (float spacing) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftext->setspacing() sets the current font spacing to `spacingspacing`. Default is 1.0. 0 is all of the letters written at the same point. This doesn't really work that well because it inflates the advance across the letter, doesn't add the same amount of spacing between the letters. I should try and explain that better, prolly. Or just fix the damn thing to do constant spacing. This was really just a way to figure out how letter advances work, anyway.. So nyah.

SWFTextField (PHP 4 >= 4.0.5)

Creates a text field object

new **swftextfield** ([int flags]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield() creates a new text field object. Text Fields are less flexible than swftext() objects- they can't be rotated, scaled non-proportionally, or skewed, but they can be used as form entries, and they can use browser-defined fonts.

The optional flags change the text field's behavior. It has the following possibles values :

- SWFTEXTFIELD_DRAWBOX draws the outline of the textfield
- SWFTEXTFIELD_HASLENGTH
- SWFTEXTFIELD_HTML allows text markup using HTML-tags
- SWFTEXTFIELD_MULTILINE allows multiple lines
- SWFTEXTFIELD_NOEDIT indicates that the field shouldn't be user-editable
- SWFTEXTFIELD_NOSELECT makes the field non-selectable
- SWFTEXTFIELD_PASSWORD obscures the data entry
- SWFTEXTFIELD_WORDWRAP allows text to wrap

Flags are combined with the bitwise OR operation. For example,

```
<?php
$t = newSWFTextField(SWFTEXTFIELD_PASSWORD | SWFTEXTFIELD_NOEDIT);
?>
```

creates a totally useless non-editable password field.

SWFTextField has the following methods : **swftextfield->setfont()**, **swftextfield->setbounds()**, **swftextfield->align()**, **swftextfield->setheight()**, **swftextfield->setleftmargin()**, **swftextfield->setrightmargin()**, **swftextfield->setmargins()**, **swftextfield->setindentation()**, **swftextfield->setlinespacing()**, **swftextfield->setcolor()**, **swftextfield->setname()** and **swftextfield->addstring()**.

SWFTextField->addstring (unknown)

Concatenates the given string to the text field

void **swftextfield->addstring** (string string) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setname() concatenates the string *string* to the text field.

SWFTextField->align (unknown)

Sets the text field alignment

void **swftextfield->align** (int alignement) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->align() sets the text field alignment to *alignement*. Valid values for *alignement* are : SWFTEXTFIELD_ALIGN_LEFT, SWFTEXTFIELD_ALIGN_RIGHT, SWFTEXTFIELD_ALIGN_CENTER and SWFTEXTFIELD_ALIGN_JUSTIFY.

SWFTextField->setbounds (unknown)

Sets the text field width and height

void **swftextfield->setbounds** (int width, int height) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setbounds() sets the text field width to *width* and height to *height*. If you don't set the bounds yourself, Ming makes a poor guess at what the bounds are.

SWFTextField->setcolor (unknown)

Sets the color of the text field.

void **swftextfield->setcolor** (int red, int green, int blue [, int a]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setcolor() sets the color of the text field. Default is fully opaque black. Color is represented using RGB system.

SWFTextField->setFont (unknown)

Sets the text field font

void **swftextfield->setfont** (string font) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setfont() sets the text field font to the [browser-defined?] *`font`* font.

SWFTextField->setHeight (unknown)

Sets the font height of this text field font.

void **swftextfield->setheight** (int height) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setheight() sets the font height of this text field font to the given height *`height`*. Default is 240.

SWFTextField->setindentation (unknown)

Sets the indentation of the first line.

void **swftextfield->setindentation** (int width) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setindentation() sets the indentation of the first line in the text field, to *width*.

SWFTextField->setLeftMargin (unknown)

Sets the left margin width of the text field.

void **swftextfield->setleftmargin** (int width) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setleftmargin() sets the left margin width of the text field to *width*. Default is 0.

SWFTextField->setLineSpacing (unknown)

Sets the line spacing of the text field.

void **swftextfield->setlinespacing** (int height) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setlinespacing() sets the line spacing of the text field to the height of *height*. Default is 40.

SWFTextField->setMargins (unknown)

Sets the margins width of the text field.

void **swftextfield->setmargins** (int left, int right) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setmargins() set both margins at once, for the man on the go.

SWFTextField->setname (unknown)

Sets the variable name

void **swftextfield->setname** (string name) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setname() sets the variable name of this text field to *name*, for form posting and action scripting purposes.

SWFTextField->setrightMargin (unknown)

Sets the right margin width of the text field.

void **swftextfield->setrightmargin** (int width) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

swftextfield->setrightmargin() sets the right margin width of the text field to *width*. Default is 0.

LIX. Miscellaneous functions

These functions were placed here because none of the other categories seemed to fit.

connection_aborted (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Returns `TRUE` if client disconnected

int **connection_aborted** (void) \linebreak

Returns `TRUE` if client disconnected. See the Connection Handling description in the Features chapter for a complete explanation.

connection_status (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Returns connection status bitfield

int **connection_status** (void) \linebreak

Returns the connection status bitfield. See the Connection Handling description in the Features chapter for a complete explanation.

connection_timeout (PHP 3>= 3.0.7, 4.0.0 - 4.0.4 only)

Return `TRUE` if script timed out

bool **connection_timeout** (void) \linebreak

Returns `TRUE` if script timed out.

Deprecated

This function is deprecated, and doesn't even exist anymore as of 4.0.5.

See the Connection Handling description in the Features chapter for a complete explanation.

constant (PHP 4 >= 4.0.4)

Returns the value of a constant

mixed **constant** (string name) \linebreak

constant() will return the value of the constant indicated by *`name`*.

constant() is useful if you need to retrieve the value of a constant, but do not know it's name. i.e. It is stored in a variable or returned by a function.

Esempio 1. constant() example

```
<?php
```

```
define ("MAXSIZE", 100);

echo MAXSIZE;
echo constant("MAXSIZE"); // same thing as the previous line

?>
```

See also define(), defined() and the section on Constants.

define (PHP 3, PHP 4 >= 4.0.0)

Defines a named constant.

bool **define** (string name, mixed value [, bool case_insensitive]) \linebreak

Defines a named constant. See the section on constants for more details.

The name of the constant is given by *name*; the value is given by *value*.

The optional third parameter *case_insensitive* is also available. If the value TRUE is given, then the constant will be defined case-insensitive. The default behaviour is case-sensitive; i.e. CONSTANT and Constant represent different values.

Esempio 1. Defining Constants

```
<?php
define ("CONSTANT", "Hello world.");
echo CONSTANT; // outputs "Hello world."
echo Constant; // outputs "Constant" and issues a notice.

define ("GREETING", "Hello you.",TRUE);
echo GREETING; // outputs "Hello you."
echo Greeting; // outputs "Hello you."

?>
```

define() returns TRUE on success and FALSE if an error occurs.

See also defined(), constant() and the section on Constants.

defined (PHP 3, PHP 4 >= 4.0.0)

Checks whether a given named constant exists

bool **defined** (string name) \linebreak

Returns `TRUE` if the named constant given by *`name`* has been defined, `FALSE` otherwise.

Esempio 1. Checking Constants

```
<?php
if (defined("CONSTANT")){ // Note that it should be quoted
    echo CONSTANT; //
    }
?>
```

See also define(), constant(), get_defined_constants() and the section on Constants.

die (unknown)

Alias of exit()

This function is an alias of exit().

eval (unknown)

Evaluate a string as PHP code

mixed **eval** (string code_str) \linebreak

eval() evaluates the string given in *`code_str`* as PHP code. Among other things, this can be useful for storing code in a database text field for later execution.

There are some factors to keep in mind when using **eval()**. Remember that the string passed must be valid PHP code, including things like terminating statements with a semicolon so the parser doesn't die on the line after the **eval()**, and properly escaping things in *`code_str`*.

Also remember that variables given values under **eval()** will retain these values in the main script afterwards.

A `return` statement will terminate the evaluation of the string immediately. In PHP 4, **eval()** returns `FALSE` unless return() is called in the evaluated code, in which case the value passed to return() is returned. In PHP 3, **eval()** does not return a value.

Esempio 1. eval() example - simple text merge

```
<?php
$string = 'cup';
$name = 'coffee';
$str = 'This is a $string with my $name in it.<br>';
```

```
echo $str;
eval ("\$str = \"$str\";");
echo $str;
?>
```

The above example will show:

```
This is a $string with my $name in it.
This is a cup with my coffee in it.
```

Suggerimento: Come con qualsiasi cosa che invia il risultato direttamente al browser, è possibile utilizzare la funzione output-control per catturare l'uscita di questa funzione e salvarla - per esempio - in una stringa.

exit (unknown)

Output a message and terminate the current script

void **exit** ([string status]) \linebreak void **exit** (int status) \linebreak

Nota: This is not a real function, but a language construct.

The **exit()** function terminates execution of the script. It prints *status* just before exiting.

If *status* is an integer, that value will also be used as the exit status. Exit statuses should be in the range 1 to 254, the exit status 255 is reserved by PHP and shall not be used.

Nota: The current CVS version does NOT print the *status* if it is an integer.

Nota: The die() function is an alias for **exit()**.

Esempio 1. exit() example

```
<?php

$filename = '/path/to/data-file';
$file = fopen ($filename, 'r')
```

```
    or exit("unable to open file ($filename)");

?>
```

get_browser (PHP 3, PHP 4 >= 4.0.0)

Tells what the user's browser is capable of

object **get_browser** ([string user_agent]) \linebreak

get_browser() attempts to determine the capabilities of the user's browser. This is done by looking up the browser's information in the `browscap.ini` file. By default, the value of $HTTP_USER_AGENT is used; however, you can alter this (i.e., look up another browser's info) by passing the optional *`user_agent`* parameter to **get_browser()**.

The information is returned in an object, which will contain various data elements representing, for instance, the browser's major and minor version numbers and ID string; `TRUE`/false values for features such as frames, JavaScript, and cookies; and so forth.

While `browscap.ini` contains information on many browsers, it relies on user updates to keep the database current. The format of the file is fairly self-explanatory.

The following example shows how one might list all available information retrieved about the user's browser.

Esempio 1. get_browser() example

```
<?php
function list_array ($array) {
    while (list ($key, $value) = each ($array)) {
    $str .= "<b>$key:</b> $value<br>\n";
    }
    return $str;
}
echo "$HTTP_USER_AGENT<hr>\n";
$browser = get_browser();
echo list_array ((array) $browser);
?>
```

The output of the above script would look something like this:

```
Mozilla/4.5 [en] (X11; U; Linux 2.2.9 i586)<hr>
<b>browser_name_pattern:</b> Mozilla/4\.5.*<br>
<b>parent:</b> Netscape 4.0<br>
<b>platform:</b> Unknown<br>
<b>majorver:</b> 4<br>
```

```
<b>minorver:</b> 5<br>
<b>browser:</b> Netscape<br>
<b>version:</b> 4<br>
<b>frames:</b> 1<br>
<b>tables:</b> 1<br>
<b>cookies:</b> 1<br>
<b>backgroundsounds:</b> <br>
<b>vbscript:</b> <br>
<b>javascript:</b> 1<br>
<b>javaapplets:</b> 1<br>
<b>activexcontrols:</b> <br>
<b>beta:</b> <br>
<b>crawler:</b> <br>
<b>authenticodeupdate:</b> <br>
<b>msn:</b> <br>
```

In order for this to work, your browscap configuration file setting must point to the correct location of the `browscap.ini` file.

For more information (including locations from which you may obtain a `browscap.ini` file), check the PHP FAQ at http://www.php.net/FAQ.php.

highlight_file (PHP 4 >= 4.0.0)

Syntax highlighting of a file

mixed **highlight_file** (string filename [, bool return]) \linebreak

The **highlight_file()** function prints out a syntax higlighted version of the code contained in *filename* using the colors defined in the built-in syntax highlighter for PHP.

If the second parameter *return* is set to TRUE then **highlight_file()** will return the highlighted code as a string instead of printing it out. If the second parameter is not set to TRUE then **highlight_file()** will return TRUE on success, FALSE on failure.

> **Nota:** The *return* parameter became available in PHP 4.2.0. Before this time it behaved like the default, which is FALSE

> **Nota:** Care should be taken when using the show_source() and **highlight_file()** functions to make sure that you do not inadvertently reveal sensitive information such as passwords or any other type of information that might create a potential security risk.

> **Nota:** Since PHP 4.2.1 this function is also affected by safe_mode and open_basedir.

Esempio 1. Creating a source highlighting URL

To setup a URL that can code hightlight any script that you pass to it, we will make use of the "ForceType" directive in Apache to generate a nice URL pattern, and use the function **highlight_file()** to show a nice looking code list.

In your httpd.conf you can add the following:

```
<Location /source>
    ForceType application/x-httpd-php
</Location>
```

And then make a file named "source" and put it in your web root directory.

```
<HTML>
<HEAD>
<TITLE>Source Display</TITLE>
</HEAD>
<BODY BGCOLOR="white">
<?php
    $script = getenv ("PATH_TRANSLATED");
    if(!$script) {
    echo "<BR><B>ERROR: Script Name needed</B><BR>";
    } else {
    if (ereg("(\.php|\.inc)$",$script)) {
    echo "<H1>Source of: $PATH_INFO</H1>\n<HR>\n";
    highlight_file($script);
    } else {
    echo "<H1>ERROR: Only PHP or include script names are allowed</H1>";
    }
    }
    echo "<HR>Processed: ".date("Y/M/d H:i:s",time());
?>
</BODY>
</HTML>
```

Then you can use an URL like the one below to display a colorized version of a script located in "/path/to/script.php" in your web site.

```
http://your.server.com/source/path/to/script.php
```

See also highlight_string(), show_source().

highlight_string (PHP 4 >= 4.0.0)

Syntax highlighting of a string

mixed **highlight_string** (string str [, bool return]) \linebreak

The **highlight_string()** function outputs a syntax highlighted version of *str* using the colors defined in the built-in syntax highlighter for PHP.

If the second parameter *return* is set to TRUE then **highlight_string()** will return the highlighted code as a string instead of printing it out. If the second parameter is not set to TRUE then **highlight_string()** will return TRUE on success, FALSE on failure.

> **Nota:** The *return* parameter became available in PHP 4.2.0. Before this time it behaved like the default, which is FALSE

See also highlight_file(), and show_source().

ignore_user_abort (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Set whether a client disconnect should abort script execution

int **ignore_user_abort** ([int setting]) \linebreak

This function sets whether a client disconnect should cause a script to be aborted. It will return the previous setting and can be called without an argument to not change the current setting and only return the current setting. See the Connection Handling section in the Features chapter for a complete description of connection handling in PHP.

iptcparse (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Parse a binary IPTC http://www.iptc.org/ block into single tags.

array **iptcparse** (string iptcblock) \linebreak

This function parses a binary IPTC block into its single tags. It returns an array using the tagmarker as an index and the value as the value. It returns FALSE on error or if no IPTC data was found. See **GetImageSize()** for a sample.

leak (PHP 3, PHP 4 >= 4.0.0)

Leak memory

void **leak** (int bytes) \linebreak

leak() leaks the specified amount of memory.

This is useful when debugging the memory manager, which automatically cleans up "leaked" memory when each request is completed.

pack (PHP 3, PHP 4 >= 4.0.0)

Pack data into binary string.

string **pack** (string format [, mixed args]) \linebreak

Pack given arguments into binary string according to `format`. Returns binary string containing data.

The idea to this function was taken from Perl and all formatting codes work the same as there, however, there are some formatting codes that are missing such as Perl's "u" format code. The format string consists of format codes followed by an optional repeater argument. The repeater argument can be either an integer value or * for repeating to the end of the input data. For a, A, h, H the repeat count specifies how many characters of one data argument are taken, for @ it is the absolute position where to put the next data, for everything else the repeat count specifies how many data arguments are consumed and packed into the resulting binary string. Currently implemented are

- a NUL-padded string
- A SPACE-padded string
- h Hex string, low nibble first
- H Hex string, high nibble first
- c signed char
- C unsigned char
- s signed short (always 16 bit, machine byte order)
- S unsigned short (always 16 bit, machine byte order)
- n unsigned short (always 16 bit, big endian byte order)
- v unsigned short (always 16 bit, little endian byte order)
- i signed integer (machine dependent size and byte order)
- I unsigned integer (machine dependent size and byte order)
- l signed long (always 32 bit, machine byte order)
- L unsigned long (always 32 bit, machine byte order)
- N unsigned long (always 32 bit, big endian byte order)
- V unsigned long (always 32 bit, little endian byte order)
- f float (machine dependent size and representation)
- d double (machine dependent size and representation)
- x NUL byte
- X Back up one byte
- @ NUL-fill to absolute position

Esempio 1. pack() format string

```
$binarydata = pack ("nvc*", 0x1234, 0x5678, 65, 66);
```

The resulting binary string will be 6 bytes long and contain the byte sequence 0x12, 0x34, 0x78, 0x56, 0x41, 0x42.

Note that the distinction between signed and unsigned values only affects the function unpack(), where as function **pack()** gives the same result for signed and unsigned format codes.

Also note that PHP internally stores integer values as signed values of a machine dependent size. If you give it an unsigned integer value too large to be stored that way it is converted to a float which often yields an undesired result.

show_source (PHP 4 >= 4.0.0)

Syntax highlighting of a file

bool **show_source** (string filename [, bool return]) \linebreak

This function is an alias to highlight_file(). For more information see the documentation there.

See also highlight_string() and highlight_file().

sleep (PHP 3, PHP 4 >= 4.0.0)

Delay execution

void **sleep** (int seconds) \linebreak

The sleep function delays program execution for the given number of *seconds*.

See also usleep() and set_time_limit()

uniqid (PHP 3, PHP 4 >= 4.0.0)

Generate a unique id

string **uniqid** (string prefix [, bool lcg]) \linebreak

uniqid() returns a prefixed unique identifier based on the current time in microseconds. The prefix can be useful for instance if you generate identifiers simultaneously on several hosts that might happen to generate the identifier at the same microsecond. *Prefix* can be up to 114 characters long.

If the optional *lcg* parameter is TRUE, **uniqid()** will add additional "combined LCG" entropy at the end of the return value, which should make the results more unique.

With an empty *prefix*, the returned string will be 13 characters long. If *lcg* is TRUE, it will be 23 characters.

> **Nota:** The *lcg* parameter is only available in PHP 4 and PHP 3.0.13 and later.

If you need a unique identifier or token and you intend to give out that token to the user via the network (i.e. session cookies), it is recommended that you use something along the lines of

```
$token = md5(uniqid("")); // no prefix
$better_token = md5(uniqid(rand(),1)); // better, difficult to guess
```

This will create a 32 character identifier (a 128 bit hex number) that is extremely difficult to predict.

unpack (PHP 3, PHP 4 >= 4.0.0)

Unpack data from binary string

array **unpack** (string format, string data) \linebreak

unpack() from binary string into array according to *format*. Returns array containing unpacked elements of binary string.

unpack() works slightly different from Perl as the unpacked data is stored in an associative array. To accomplish this you have to name the different format codes and separate them by a slash /.

Esempio 1. unpack() format string

```
$array = unpack ("c2chars/nint", $binarydata);
```

The resulting array will contain the entries "chars1", "chars2" and "int".

For an explanation of the format codes see also: pack()

Note that PHP internally stores integral values as signed. If you unpack a large unsigned long and it is of the same size as PHP internally stored values the result will be a negative number even though unsigned unpacking was specified.

usleep (PHP 3, PHP 4 >= 4.0.0)

Delay execution in microseconds

void **usleep** (int micro_seconds) \linebreak

The **usleep()** function delays program execution for the given number of *`micro_seconds`*. A microsecond is one millionth of a second.

See also sleep() and set_time_limit().

> **Nota:** This function does not work on Windows systems.

LX. mnoGoSearch Functions

These functions allow you to access mnoGoSearch (former UdmSearch) free search engine. In order to have these functions available, you must compile php with mnogosearch support by using the `--with-mnogosearch` option. If you use this option without specifying the path to mnogosearch, php will look for mnogosearch under /usr/local/mnogosearch path by default. If you installed mnogosearch at other path you should specify it: `--with-mnogosearch=DIR`.

mnoGoSearch is a full-featured search engine software for intranet and internet servers, distributed under the GNU license. mnoGoSearch has number of unique features, which makes it appropriate for a wide range of application from search within your site to a specialized search system such as cooking recipes or newspaper search, ftp archive search, news articles search, etc. It offers full-text indexing and searching for HTML, PDF, and text documents. mnoGoSearch consists of two parts. The first is an indexing mechanism (indexer). The purpose of indexer is to walk through HTTP, FTP, NEWS servers or local files, recursively grabbing all the documents and storing meta-data about that documents in a SQL database in a smart and effective manner. After every document is referenced by its corresponding URL, meta-data collected by indexer is used later in a search process. The search is performed via Web interface. C CGI, PHP and Perl search front ends are included.

> **Nota:** php contains built-in mysql access library, which can be used to access mysql. It is known that mnoGoSearch is not compatible with this built-in library and can work only with generic mysql libraries. Thus, if you use mnoGoSearch with mysql, during php configuration you have to indicate directory of mysql installation, that was used during mnoGoSearch configuration, i.e. for example: `--with-mnogosearch --with-mysql=/usr`.

You need at least 3.1.10 version of mnoGoSearch installed to use these functions.

More information about mnoGoSearch can be found at http://www.mnogosearch.ru/.

udm_add_search_limit (PHP 4 >= 4.0.5)

Add various search limits

int **udm_add_search_limit** (int agent, int var, string val) \linebreak

udm_add_search_limit() returns `TRUE` on success, `FALSE` on error. Adds search restrictions.

`agent` - a link to Agent, received after call to udm_alloc_agent().

`var` - defines parameter, indicating limit.

`val` - defines value of the current parameter.

Possible *`var`* values:

- UDM_LIMIT_URL - defines document URL limitations to limit search through subsection of database. It supports SQL % and _ LIKE wildcards, where % matches any number of characters, even zero characters, and _ matches exactly one character. E.g. http://my.domain.__/catalog may stand for http://my.domain.ru/catalog and http://my.domain.ua/catalog.
- UDM_LIMIT_TAG - defines site TAG limitations. In indexer-conf you can assign specific TAGs to various sites and parts of a site. Tags in mnoGoSearch 3.1.x are lines, that may contain metasymbols % and _. Metasymbols allow searching among groups of tags. E.g. there are links with tags ABCD and ABCE, and search restriction is by ABC_ - the search will be made among both of the tags.
- UDM_LIMIT_LANG - defines document language limitations.
- UDM_LIMIT_CAT - defines document category limitations. Categories are similar to tag feature, but nested. So you can have one category inside another and so on. You have to use two characters for each level. Use a hex number going from 0-F or a 36 base number going from 0-Z. Therefore a top-level category like 'Auto' would be 01. If it has a subcategory like 'Ford', then it would be 01 (the parent category) and then 'Ford' which we will give 01. Put those together and you get 0101. If 'Auto' had another subcategory named 'VW', then it's id would be 01 because it belongs to the 'Ford' category and then 02 because it's the next category. So it's id would be 0102. If VW had a sub category called 'Engine' then it's id would start at 01 again and it would get the 'VW' id 02 and 'Auto' id of 01, making it 010201. If you want to search for sites under that category then you pass it cat=010201 in the url.
- UDM_LIMIT_DATE - defines limitation by date document was modified.

  Format of parameter value: a string with first character < or >, then with no space - date in unixtime format, for example:

  Udm_Add_Search_Limit($udm,UDM_LIMIT_DATE,"<908012006");

  If > character is used, then search will be restricted to those documents having modification date greater than entered. If <, then smaller.

udm_alloc_agent (PHP 4 >= 4.0.5)

Allocate mnoGoSearch session

int **udm_alloc_agent** (string dbaddr [, string dbmode]) \linebreak

udm_alloc_agent() returns mnogosearch agent identifier on success, `FALSE` on error. This function creates a session with database parameters.

`dbaddr` - URL-style database description. Options (type, host, database name, port, user and password) to connect to SQL database. Do not matter for built-in text files support. Format: DBAddr DBType:[//[DBUser[:DBPass]@]DBHost[:DBPort]]/DBName/ Currently supported DBType values are: mysql, pgsql, msql, solid, mssql, oracle, ibase. Actually, it does not matter for native libraries support. But ODBC users should specify one of supported values. If your database type is not supported, you may use "unknown" instead.

`dbmode` - You may select SQL database mode of words storage. When "single" is specified, all words are stored in the same table. If "multi" is selected, words will be located in different tables depending of their lengths. "multi" mode is usually faster but requires more tables in database. If "crc" mode is selected, mnoGoSearch will store 32 bit integer word IDs calculated by CRC32 algorythm instead of words. This mode requres less disk space and it is faster comparing with "single" and "multi" modes. "crc-multi" uses the same storage structure with the "crc" mode, but also stores words in different tables depending on words lengths like "multi" mode. Format: DBMode single/multi/crc/crc-multi

> **Nota:** *`dbaddr`* and *`dbmode`* must match those used during indexing.

> **Nota:** In fact this function does not open connection to database and thus does not check entered login and password. Actual connection to database and login/password verification is done by udm_find().

udm_api_version (PHP 4 >= 4.0.5)

Get mnoGoSearch API version.

int **udm_api_version** (void) \linebreak

udm_api_version() returns mnoGoSearch API version number. E.g. if mnoGoSearch 3.1.10 API is used, this function will return `30110`.

This function allows user to identify which API functions are available, e.g. udm_get_doc_count() function is only available in mnoGoSearch 3.1.11 or later.

Example:

```
if (udm_api_version() >= 30111) {
print  "Total number of urls in database: ".udm_get_doc_count($udm)."<br>\n";
}
```

udm_cat_list (PHP 4 >= 4.0.6)

Get all the categories on the same level with the current one.

array **udm_cat_list** (int agent, string category) \linebreak

udm_cat_list() returns array listing all categories of the same level as current category in the categories tree.

The function can be useful for developing categories tree browser.

Returns array with the following format:

The array consists of pairs. Elements with even index numbers contain category paths, odd elements contain corresponding category names.

```
 $array[0] will contain '020300'
 $array[1] will contain 'Audi'
 $array[2] will contain '020301'
 $array[3] will contain 'BMW'
 $array[4] will contain '020302'
 $array[5] will contain 'Opel'
 ...
etc.
```

Following is an example of displaying links of the current level in format:
Audi
BMW
Opel
...

```
<?php
 $cat_list_arr = udm_cat_list($udm_agent,$cat);
 $cat_list = ”;
 for ($i=0; $i<count($cat_list_arr); $i+=2) {
   $path = $cat_list_arr[$i];
   $name = $cat_list_arr[$i+1];
   $cat_list .= "<a href=\"$PHP_SELF?cat=$path\">$name</a><br>";
 }
?>
```

udm_cat_path (PHP 4 >= 4.0.6)

Get the path to the current category.

array **udm_cat_path** (int agent, string category) \linebreak

udm_cat_path() returns array describing path in the categories tree from the tree root to the current category.

agent - agent link identifier.

category - current category - the one to get path to.

Returns array with the following format:

The array consists of pairs. Elements with even index numbers contain category paths, odd elements contain corresponding category names.

For example, the call `$array=udm_cat_path($agent, '02031D');` may return the following array:

$array[0] will contain ''
$array[1] will contain 'Root'
$array[2] will contain '02'
$array[3] will contain 'Sport'
$array[4] will contain '0203'
$array[5] will contain 'Auto'
$array[4] will contain '02031D'
$array[5] will contain 'Ferrari'

Esempio 1. Specifying path to the current category in the following format: '> Root > Sport > Auto > Ferrari'

```
<?php
  $cat_path_arr = udm_cat_path($udm_agent,$cat);
  $cat_path = '';
  for ($i=0; $i<count($cat_path_arr); $i+=2) {
    $path = $cat_path_arr[$i];
    $name = $cat_path_arr[$i+1];
    $cat_path .= " > <a href=\"$PHP_SELF?cat=$path\">$name</a> ";
  }
?>
```

udm_check_charset (PHP 4 >= 4.2.0)

Check if the given charset is known to mnogosearch

int **udm_check_charset** (int agent, string charset) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

udm_check_stored (PHP 4 >= 4.2.0)

Check connection to stored

int **udm_check_stored** (int agent, int link, string doc_id) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

udm_clear_search_limits (PHP 4 >= 4.0.5)

Clear all mnoGoSearch search restrictions

int **udm_clear_search_limits** (int agent) \linebreak

udm_clear_search_limits() resets defined search limitations and returns `TRUE`.

udm_close_stored (PHP 4 >= 4.2.0)

Close connection to stored

int **udm_close_stored** (int agent, int link) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

udm_crc32 (PHP 4 >= 4.2.0)

Return CRC32 checksum of gived string

int **udm_crc32** (int agent, string str) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

udm_errno (PHP 4 >= 4.0.5)

Get mnoGoSearch error number

int **udm_errno** (int agent) \linebreak

udm_errno() returns mnoGoSearch error number, zero if no error.

agent - link to agent identifier, received after call to udm_alloc_agent().

Receiving numeric agent error code.

udm_error (PHP 4 >= 4.0.5)

Get mnoGoSearch error message

string **udm_error** (int agent) \linebreak

udm_error() returns mnoGoSearch error message, empty string if no error.

agent - link to agent identifier, received after call to udm_alloc_agent().

Receiving agent error message.

udm_find (PHP 4 >= 4.0.5)

Perform search

int **udm_find** (int agent, string query) \linebreak

udm_find() returns result link identifier on success, FALSE on error.

The search itself. The first argument - session, the next one - query itself. To find something just type words you want to find and press SUBMIT button. For example, "mysql odbc". You should not use quotes " in query, they are written here only to divide a query from other text. mnoGoSearch will find all documents that contain word "mysql" and/or word "odbc". Best documents having bigger weights will be displayed first. If you use search mode ALL, search will return documents that

contain both (or more) words you entered. In case you use mode ANY, the search will return list of documents that contain any of the words you entered. If you want more advanced results you may use query language. You should select "bool" match mode in the search from.

mnoGoSearch understands the following boolean operators:

& - logical AND. For example, "mysql & odbc". mnoGoSearch will find any URLs that contain both "mysql" and "odbc".

| - logical OR. For example "mysql|odbc". mnoGoSearch will find any URLs, that contain word "mysql" or word "odbc".

~ - logical NOT. For example "mysql & ~odbc". mnoGoSearch will find URLs that contain word "mysql" and do not contain word "odbc" at the same time. Note that ~ just excludes given word from results. Query "~odbc" will find nothing!

() - group command to compose more complex queries. For example "(mysql | msql) & ~postgres". Query language is simple and powerful at the same time. Just consider query as usual boolean expression.

udm_free_agent (PHP 4 >= 4.0.5)

Free mnoGoSearch session

int **udm_free_agent** (int agent) \linebreak

udm_free_agent() returns `TRUE` on success, `FALSE` on error.

`agent` - link to agent identifier, received ‘ after call to udm_alloc_agent().

Freeing up memory allocated for agent session.

udm_free_ispell_data (PHP 4 >= 4.0.5)

Free memory allocated for ispell data

int **udm_free_ispell_data** (int agent) \linebreak

udm_free_ispell_data() always returns `TRUE`.

`agent` - agent link identifier, received after call to udm_alloc_agent().

> **Nota:** This function is supported beginning from version 3.1.12 of mnoGoSearch and it does not do anything in previous versions.

udm_free_res (PHP 4 >= 4.0.5)

Free mnoGoSearch result

int **udm_free_res** (int res) \linebreak

udm_free_res() returns TRUE on success, FALSE on error.

res - a link to result identifier, received after call to udm_find().

Freeing up memory allocated for results.

udm_get_doc_count (PHP 4 >= 4.0.5)

Get total number of documents in database.

int **udm_get_doc_count** (int agent) \linebreak

udm_get_doc_count() returns number of documents in database.

agent - link to agent identifier, received after call to udm_alloc_agent().

> **Nota:** This function is supported only in mnoGoSearch 3.1.11 or later.

udm_get_res_field (PHP 4 >= 4.0.5)

Fetch mnoGoSearch result field

string **udm_get_res_field** (int res, int row, int field) \linebreak

udm_get_res_field() returns result field value on success, FALSE on error.

res - a link to result identifier, received after call to udm_find().

row - the number of the link on the current page. May have values from 0 to *UDM_PARAM_NUM_ROWS-1*.

field - field identifier, may have the following values:

- UDM_FIELD_URL - document URL field
- UDM_FIELD_CONTENT - document Content-type field (for example, text/html).
- UDM_FIELD_CATEGORY - document category field. Use udm_cat_path() to get full path to current category from the categories tree root. (This parameter is available only in PHP 4.0.6 or later).
- UDM_FIELD_TITLE - document title field.
- UDM_FIELD_KEYWORDS - document keywords field (from META KEYWORDS tag).
- UDM_FIELD_DESC - document description field (from META DESCRIPTION tag).
- UDM_FIELD_TEXT - document body text (the first couple of lines to give an idea of what the document is about).
- UDM_FIELD_SIZE - document size.
- UDM_FIELD_URLID - unique URL ID of the link.
- UDM_FIELD_RATING - page rating (as calculated by mnoGoSearch).
- UDM_FIELD_MODIFIED - last-modified field in unixtime format.

- UDM_FIELD_ORDER - the number of the current document in set of found documents.
- UDM_FIELD_CRC - document CRC.

udm_get_res_param (PHP 4 >= 4.0.5)

Get mnoGoSearch result parameters

string **udm_get_res_param** (int res, int param) \linebreak

udm_get_res_param() returns result parameter value on success, `FALSE` on error.

`res` - a link to result identifier, received after call to udm_find().

`param` - parameter identifier, may have the following values:

- UDM_PARAM_NUM_ROWS - number of received found links on the current page. It is equal to UDM_PARAM_PAGE_SIZE for all search pages, on the last page - the rest of links.
- UDM_PARAM_FOUND - total number of results matching the query.
- UDM_PARAM_WORDINFO - information on the words found. E.g. search for "a good book" will return "a: stopword, good:5637, book: 120"
- UDM_PARAM_SEARCHTIME - search time in seconds.
- UDM_PARAM_FIRST_DOC - the number of the first document displayed on current page.
- UDM_PARAM_LAST_DOC - the number of the last document displayed on current page.

udm_load_ispell_data (PHP 4 >= 4.0.5)

Load ispell data

int **udm_load_ispell_data** (int agent, int var, string val1, string val2, int flag) \linebreak

udm_load_ispell_data() loads ispell data. Returns `TRUE` on success, `FALSE` on error.

`agent` - agent link identifier, received after call to udm_alloc_agent().

`var` - parameter, indicating the source for ispell data. May have the following values:

After using this function to free memory allocated for ispell data, please use udm_free_ispell_data(), even if you use UDM_ISPELL_TYPE_SERVER mode.

The fastest mode is UDM_ISPELL_TYPE_SERVER. UDM_ISPELL_TYPE_TEXT is slower and UDM_ISPELL_TYPE_DB is the slowest. The above pattern is `TRUE` for mnoGoSearch 3.1.10 - 3.1.11. It is planned to speed up DB mode in future versions and it is going to be faster than TEXT mode.

- UDM_ISPELL_TYPE_DB - indicates that ispell data should be loaded from SQL. In this case, parameters *`val1`* and *`val2`* are ignored and should be left blank. *`flag`* should be equal to `1`.

> **Nota:** *`flag`* indicates that after loading ispell data from defined source it sould be sorted (it is necessary for correct functioning of ispell). In case of loading ispell data from files there may be several calls to **udm_load_ispell_data()**, and there is no sense to sort data after every

call, but only after the last one. Since in db mode all the data is loaded by one call, this parameter should have the value `1`. In this mode in case of error, e.g. if ispell tables are absent, the function will return `FALSE` and code and error message will be accessible through udm_error() and udm_errno().

Example:

```
if (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_DB,”,”,1)) {
  printf("Error #%d: ’%s’\n", udm_errno($udm), udm_error($udm));
  exit;
}
```

- UDM_ISPELL_TYPE_AFFIX - indicates that ispell data should be loaded from file and initiates loading affixes file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return `FALSE`, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

  Example:

```
if ((! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_AFFIX,’en’,’/opt/ispell/en.aff’,0
   (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_AFFIX,’ru’,’/opt/ispell/ru.aff’,0)
   (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SPELL,’en’,’/opt/ispell/en.dict’,0
   (! udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SPELL,’ru’,’/opt/ispell/ru.dict’,1
exit;
}
```

  Nota: *flag* is equal to `1` only in the last call.

- UDM_ISPELL_TYPE_SPELL - indicates that ispell data should be loaded from file and initiates loading of ispell dictionary file. In this case *val1* defines double letter language code for which affixes are loaded, and *val2* - file path. Please note, that if a relative path entered, the module looks for the file not in UDM_CONF_DIR, but in relation to current path, i.e. to the path where the script is executed. In case of error in this mode, e.g. if file is absent, the function will return `FALSE`, and an error message will be displayed. Error message text cannot be accessed through udm_error() and udm_errno(), since those functions can only return messages associated with SQL. Please, see *flag* parameter description in UDM_ISPELL_TYPE_DB.

```
if ((! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,’en’,’/opt/ispell/en.aff’,0
```

```
  (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_AFFIX,'ru','/opt/ispell/ru.aff',0)
  (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'en','/opt/ispell/en.dict',0
  (! Udm_Load_Ispell_Data($udm,UDM_ISPELL_TYPE_SPELL,'ru','/opt/ispell/ru.dict',1
exit;
}
```

Nota: *flag* is equal to `1` only in the last call.

- UDM_ISPELL_TYPE_SERVER - enables spell server support. *val1* parameter indicates address of the host running spell server. *val2* ' is not used yet, but in future releases it is going to indicate number of port used by spell server. *flag* parameter in this case is not needed since ispell data is stored on spellserver already sorted.

  Spelld server reads spell-data from a separate configuration file (/usr/local/mnogosearch/etc/spelld.conf by default), sorts it and stores in memory. With clients server communicates in two ways: to indexer all the data is transferred (so that indexer starts faster), from search.cgi server receives word to normalize and then passes over to client (search.cgi) list of normalized word forms. This allows fastest, compared to db and text modes processing of search queries (by omitting loading and sorting all the spell data).

  udm_load_ispell_data() function in UDM_ISPELL_TYPE_SERVER mode does not actually load ispell data, but only defines server address. In fact, server is automatically used by udm_find() function when performing search. In case of errors, e.g. if spellserver is not running or invalid host indicated, there are no messages returned and ispell conversion does not work.

  Nota: This function is available in mnoGoSearch 3.1.12 or later.

  Example:

```
if (!udm_load_ispell_data($udm,UDM_ISPELL_TYPE_SERVER,'','',1)) {
    printf("Error loading ispell data from server<br>\n");
    exit;
}
```

udm_open_stored (PHP 4 >= 4.2.0)

Open connection to stored

int **udm_open_stored** (int agent, string storedaddr) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

udm_set_agent_param (PHP 4 >= 4.0.5)

Set mnoGoSearch agent session parameters

int **udm_set_agent_param** (int agent, int var, string val) \linebreak

udm_set_agent_param() returns TRUE on success, FALSE on error. Defines mnoGoSearch session parameters.

The following parameters and their values are available:

- UDM_PARAM_PAGE_NUM - used to choose search results page number (results are returned by pages beginning from 0, with UDM_PARAM_PAGE_SIZE results per page).
- UDM_PARAM_PAGE_SIZE - number of search results displayed on one page.
- UDM_PARAM_SEARCH_MODE - search mode. The following values available: UDM_MODE_ALL - search for all words; UDM_MODE_ANY - search for any word; UDM_MODE_PHRASE - phrase search; UDM_MODE_BOOL - boolean search. See udm_find() for details on boolean search.
- UDM_PARAM_CACHE_MODE - turns on or off search result cache mode. When enabled, the search engine will store search results to disk. In case a similar search is performed later, the engine will take results from the cache for faster performance. Available values: UDM_CACHE_ENABLED, UDM_CACHE_DISABLED.
- UDM_PARAM_TRACK_MODE - turns on or off trackquery mode. Since version 3.1.2 mnoGoSearch has a query tracking support. Note that tracking is implemented in SQL version only and not available in built-in database. To use tracking, you have to create tables for tracking support. For MySQL, use create/mysql/track.txt. When doing a search, front-end uses those tables to store query words, a number of found documents and current UNIX timestamp in seconds. Available values: UDM_TRACK_ENABLED, UDM_TRACK_DISABLED.
- UDM_PARAM_PHRASE_MODE - defines whether index database using phrases ("phrase" parameter in indexer.conf). Possible values: UDM_PHRASE_ENABLED and UDM_PHRASE_DISABLED. Please note, that if phrase search is enabled (UDM_PHRASE_ENABLED), it is still possible to do search in any mode (ANY, ALL, BOOL or PHRASE). In 3.1.10 version of mnoGoSearch phrase search is supported only in sql and built-in database modes, while beginning with 3.1.11 phrases are supported in cachemode as well.

  Examples of phrase search:

  "Arizona desert" - This query returns all indexed documents that contain "Arizona desert" as a phrase. Notice that you need to put double quotes around the phrase

- UDM_PARAM_CHARSET - defines local charset. Available values: set of charsets supported by mnoGoSearch, e.g. koi8-r, cp1251, ...

- UDM_PARAM_STOPFILE - Defines name and path to stopwords file. (There is a small difference with mnoGoSearch - while in mnoGoSearch if relative path or no path entered, it looks for this file in relation to UDM_CONF_DIR, the module looks for the file in relation to current path, i.e. to the path where the php script is executed.)
- UDM_PARAM_STOPTABLE - Load stop words from the given SQL table. You may use several StopwordTable commands. This command has no effect when compiled without SQL database support.
- UDM_PARAM_WEIGHT_FACTOR - represents weight factors for specific document parts. Currently body, title, keywords, description, url are supported. To activate this feature please use degrees of 2 in *Weight commands of the indexer.conf. Let's imagine that we have these weights:

  URLWeight   1
  BodyWeight   2
  TitleWeight   4
  KeywordWeight 8
  DescWeight   16

  As far as indexer uses bit OR operation for word weights when some word presents several time in the same document, it is possible at search time to detect word appearance in different document parts. Word which appears only in the body will have 00000010 argegate weight (in binary notation). Word used in all document parts will have 00011111 aggregate weight.

  This parameter's value is a string of hex digits ABCDE. Each digit is a factor for corresponding bit in word weight. For the given above weights configuration:

  E is a factor for weight 1 (URL Weight bit)
  D is a factor for weight 2 (BodyWeight bit)
  C is a factor for weight 4 (TitleWeight bit)
  B is a factor for weight 8 (KeywordWeight bit)
  A is a factor for weight 16 (DescWeight bit)

  Examples:

  UDM_PARAM_WEIGHT_FACTOR=00001 will search through URLs only.

  UDM_PARAM_WEIGHT_FACTOR=00100 will search through Titles only.

  UDM_PARAM_WEIGHT_FACTOR=11100 will search through Title,Keywords,Description but not through URL and Body.

  UDM_PARAM_WEIGHT_FACTOR=F9421 will search through:

  Description with factor 15 (F hex)
  Keywords with factor 9
  Title with factor 4
  Body with factor 2
  URL with factor 1

  If UDM_PARAM_WEIGHT_FACTOR variable is ommited, original weight value is taken to sort results. For a given above weight configuration it means that document description has a most big weight 16.

- UDM_PARAM_WORD_MATCH - word match. You may use this parameter to choose word match type. This feature works only in "single" and "multi" modes using SQL based and built-in

database. It does not work in cachemode and other modes since they use word CRC and do not support substring search. Available values:

UDM_MATCH_BEGIN - word beginning match;

UDM_MATCH_END - word ending match;

UDM_MATCH_WORD - whole word match;

UDM_MATCH_SUBSTR - word substring match.

- UDM_PARAM_MIN_WORD_LEN - defines minimal word length. Any word shorter this limit is considered to be a stopword. Please note that this parameter value is inclusive, i.e. if UDM_PARAM_MIN_WORD_LEN=3, a word 3 characters long will not be considered a stopword, while a word 2 characters long will be. Default value is 1.
- UDM_PARAM_ISPELL_PREFIXES - Possible values: UDM_PREFIXES_ENABLED and UDM_PREFIXES_DISABLED, that respectively enable or disable using prefixes. E.g. if a word "tested" is in search query, also words like "test", "testing", etc. Only suffixes are supported by default. Prefixes usually change word meanings, for example if somebody is searching for the word "tested" one hardly wants "untested" to be found. Prefixes support may also be found useful for site's spelling checking purposes. In order to enable ispell, you have to load ispell data with udm_load_ispell_data().
- UDM_PARAM_CROSS_WORDS - enables or disables crosswords support. Possible values: UDM_CROSS_WORDS_ENABLED and UDM_CROSS_WORDS_DISABLED.

  The corsswords feature allows to assign words between and also to a document this link leads to. It works in SQL database mode and is not supported in built-in database and Cachemode.

  > **Nota:** Crosswords are supported only in mnoGoSearch 3.1.11 or later.

- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default `/var` directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.
- UDM_PARAM_VARDIR - specifies a custom path to directory where indexer stores data when using built-in database and in cache mode. By default `/var` directory of mnoGoSearch installation is used. Can have only string values. The parameter is available in PHP 4.1.0 or later.

LXI. mSQL functions

These functions allow you to access mSQL database servers. In order to have these functions available, you must compile php with msql support by using the `--with-msql[=dir]` option. The default location is /usr/local/Hughes.

More information about mSQL can be found at http://www.hughes.com.au/.

msql (PHP 3, PHP 4 >= 4.0.0)

Send mSQL query

int **msql** (string database, string query, int link_identifier) \linebreak

Returns a positive mSQL query identifier to the query result, or `FALSE` on error.

msql() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the mSQL server and if no such link is found it'll try to create one as if msql_connect() was called with no arguments (see msql_connect()).

msql_affected_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Returns number of affected rows

int **msql_affected_rows** (int query_identifier) \linebreak

Returns number of affected ("touched") rows by a specific query (i.e. the number of rows returned by a SELECT, the number of rows modified by an update, or the number of rows removed by a delete).

See also: msql_query().

msql_close (PHP 3, PHP 4 >= 4.0.0)

Close mSQL connection

int **msql_close** (int link_identifier) \linebreak

Returns `TRUE` on success, `FALSE` on error.

msql_close() closes the link to a mSQL database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

msql_close() will not close persistent links generated by msql_pconnect().

See also: msql_connect() and msql_pconnect().

msql_connect (PHP 3, PHP 4 >= 4.0.0)

Open mSQL connection

int **msql_connect** ([string hostname [, string server [, string username [, string password]]]]) \linebreak

Returns a positive mSQL link identifier on success, or `FALSE` on error.

msql_connect() establishes a connection to a mSQL server. The *`server`* parameter can also include a port number. eg. "hostname:port". It defaults to 'localhost'.

In case a second call is made to **msql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling msql_close().

See also msql_pconnect(), msql_close().

msql_create_db (PHP 3, PHP 4 >= 4.0.0)

Create mSQL database

int **msql_create_db** (string database name [, int link_identifier]) \linebreak

msql_create_db() attempts to create a new database on the server associated with the specified link identifier.

See also: msql_drop_db().

msql_createdb (PHP 3, PHP 4 >= 4.0.0)

Create mSQL database

int **msql_createdb** (string database name [, int link_identifier]) \linebreak

Identical to msql_create_db().

msql_data_seek (PHP 3, PHP 4 >= 4.0.0)

Move internal row pointer

int **msql_data_seek** (int query_identifier, int row_number) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

msql_data_seek() moves the internal row pointer of the mSQL result associated with the specified query identifier to pointer to the specifyed row number. The next call to msql_fetch_row() would return that row.

See also: msql_fetch_row().

msql_dbname (PHP 3, PHP 4 >= 4.0.0)

Get current mSQL database name

string **msql_dbname** (int query_identifier, int i) \linebreak

msql_dbname() returns the database name stored in position *i* of the result pointer returned from the msql_listdbs() function. The msql_numrows() function can be used to determine how many database names are available.

msql_drop_db (PHP 3, PHP 4 >= 4.0.0)

Drop (delete) mSQL database

int **msql_drop_db** (string database_name, int link_identifier) \linebreak

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

msql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

See also: msql_create_db().

msql_dropdb (PHP 3, PHP 4 >= 4.0.0)

Drop (delete) mSQL database

See msql_drop_db().

msql_error (PHP 3, PHP 4 >= 4.0.0)

Returns error message of last msql call

string **msql_error** ([int link_identifier]) \linebreak

Errors coming back from the mSQL database backend no longer issue warnings. Instead, use these functions to retrieve the error string.

msql_fetch_array (PHP 3, PHP 4 >= 4.0.0)

Fetch row as array

int **msql_fetch_array** (int query_identifier [, int result_type]) \linebreak

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

msql_fetch_array() is an extended version of msql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

The second optional argument *result_type* in **msql_fetch_array()** is a constant and can take the following values: MSQL_ASSOC, MSQL_NUM, and MSQL_BOTH.

Be careful if you are retrieving results from a query that may return a record that contains only one field that has a value of 0 (or an empty string, or `NULL`).

An important thing to note is that using **msql_fetch_array()** is NOT significantly slower than using msql_fetch_row(), while it provides a significant added value.

For further details, also see msql_fetch_row().

msql_fetch_field (PHP 3, PHP 4 >= 4.0.0)

Get field information

object **msql_fetch_field** (int query_identifier, int field_offset) \linebreak

Returns an object containing field information

msql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **msql_fetch_field()** is retreived.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- not_null - 1 if the column cannot be `NULL`
- primary_key - 1 if the column is a primary key
- unique - 1 if the column is a unique key
- type - the type of the column

See also msql_field_seek().

msql_fetch_object (PHP 3, PHP 4 >= 4.0.0)

Fetch row as object

int **msql_fetch_object** (int query_identifier [, int result_type]) \linebreak

Returns an object with properties that correspond to the fetched row, or `FALSE` if there are no more rows.

msql_fetch_object() is similar to msql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

The optional second argument *`result_type`* in msql_fetch_array() is a constant and can take the following values: MSQL_ASSOC, MSQL_NUM, and MSQL_BOTH.

Speed-wise, the function is identical to msql_fetch_array(), and almost as quick as msql_fetch_row() (the difference is insignificant).

See also: msql_fetch_array() and msql_fetch_row().

msql_fetch_row (PHP 3, PHP 4 >= 4.0.0)

Get row as enumerated array

array **msql_fetch_row** (int query_identifier) \linebreak

Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

msql_fetch_row() fetches one row of data from the result associated with the specified query identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **msql_fetch_row()** would return the next row in the result set, or `FALSE` if there are no more rows.

See also: msql_fetch_array(), msql_fetch_object(), msql_data_seek(), and msql_result().

msql_field_seek (PHP 3, PHP 4 >= 4.0.0)

Set field offset

int **msql_field_seek** (int query_identifier, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to msql_fetch_field() won't include a field offset, this field would be returned.

See also: msql_fetch_field().

msql_fieldflags (PHP 3, PHP 4 >= 4.0.0)

Get field flags

string **msql_fieldflags** (int query_identifier, int i) \linebreak

msql_fieldflags() returns the field flags of the specified field. Currently this is either, "not `NULL`", "primary key", a combination of the two or "" (an empty string).

msql_fieldlen (PHP 3, PHP 4 >= 4.0.0)

Get field length

int **msql_fieldlen** (int query_identifier, int i) \linebreak

msql_fieldlen() returns the length of the specified field.

msql_fieldname (PHP 3, PHP 4 >= 4.0.0)

Get field name

string **msql_fieldname** (int query_identifier, int field) \linebreak

msql_fieldname() returns the name of the specified field. *query_identifier* is the query identifier, and *field* is the field index. `msql_fieldname($result, 2);` will return the name of the second field in the result associated with the result identifier.

msql_fieldtable (PHP 3, PHP 4 >= 4.0.0)

Get table name for field

int **msql_fieldtable** (int query_identifier, int field) \linebreak

Returns the name of the table *field* was fetched from.

msql_fieldtype (PHP 3, PHP 4 >= 4.0.0)

Get field type

string **msql_fieldtype** (int query_identifier, int i) \linebreak

msql_fieldtype() is similar to the msql_fieldname() function. The arguments are identical, but the field type is returned. This will be one of "int", "char" or "real".

msql_free_result (PHP 3, PHP 4 >= 4.0.0)

Free result memory

int **msql_free_result** (int query_identifier) \linebreak

msql_free_result() frees the memory associated with *query_identifier*. When PHP completes a request, this memory is freed automatically, so you only need to call this function when you want to make sure you don't use too much memory while the script is running.

msql_freeresult (PHP 3, PHP 4 >= 4.0.0)

Free result memory

See msql_free_result()

msql_list_dbs (PHP 3, PHP 4 >= 4.0.0)

List mSQL databases on server

int **msql_list_dbs** (void) \linebreak

msql_list_dbs() will return a result pointer containing the databases available from the current msql daemon. Use the msql_dbname() function to traverse this result pointer.

msql_list_fields (PHP 3, PHP 4 >= 4.0.0)

List result fields

int **msql_list_fields** (string database, string tablename) \linebreak

msql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with msql_fieldflags(), msql_fieldlen(), msql_fieldname(), and msql_fieldtype(). A query identifier is a positive integer. The function returns `-1` if a error occurs. A string describing the error will be placed in `$phperrmsg`, and unless the function was called as `@msql_list_fields()` then this error string will also be printed out.

See also msql_error().

msql_list_tables (PHP 3, PHP 4 >= 4.0.0)

List tables in an mSQL database

int **msql_list_tables** (string database) \linebreak

msql_list_tables() takes a database name and result pointer much like the msql() function. The msql_tablename() function should be used to extract the actual table names from the result pointer.

msql_listdbs (PHP 3, PHP 4 >= 4.0.0)

List mSQL databases on server

See msql_list_dbs().

msql_listfields (PHP 3, PHP 4 >= 4.0.0)

List result fields

See msql_list_fields().

msql_listtables (PHP 3, PHP 4 >= 4.0.0)

List tables in an mSQL database

See msql_list_tables().

msql_num_fields (PHP 3, PHP 4 >= 4.0.0)

Get number of fields in result

int **msql_num_fields** (int query_identifier) \linebreak

msql_num_fields() returns the number of fields in a result set.

See also: msql(), msql_query(), msql_fetch_field(), and msql_num_rows().

msql_num_rows (PHP 3, PHP 4 >= 4.0.0)

Get number of rows in result

int **msql_num_rows** (int query_identifier) \linebreak

msql_num_rows() returns the number of rows in a result set.

See also: msql(), msql_query(), and msql_fetch_row().

msql_numfields (PHP 3, PHP 4 >= 4.0.0)

Get number of fields in result

int **msql_numfields** (int query_identifier) \linebreak

Identical to msql_num_fields().

msql_numrows (PHP 3, PHP 4 >= 4.0.0)

Get number of rows in result

int **msql_numrows** (void) \linebreak

Identical to msql_num_rows().

msql_pconnect (PHP 3, PHP 4 >= 4.0.0)

Open persistent mSQL connection

int **msql_pconnect** ([string server [, string username [, string password]]]) \linebreak

Returns a positive mSQL persistent link identifier on success, or FALSE on error.

msql_pconnect() acts very much like msql_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (msql_close() will not close links established by **msql_pconnect()**).

This type of links is therefore called 'persistent'.

msql_query (PHP 3, PHP 4 >= 4.0.0)

Send mSQL query

int **msql_query** (string query, int link_identifier) \linebreak

msql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if msql_connect() was called, and use it.

Returns a positive mSQL query identifier on success, or FALSE on error.

Esempio 1. msql_query()

```
<?php
$link = msql_connect("dbserver")
   or die("unable to connect to msql server: ".msql_error());
msql_select_db("db", $link)
   or die("unable to select database 'db': ".msql_error());

$result = msql_query("SELECT * FROM table WHERE id=1", $link);
if (!$result) {
   die("query failed: ".msql_error());
}

while ($row = msql_fetch_array($result)) {
    echo $row["id"];
}
?>
```

See also: msql(), msql_select_db(), and msql_connect().

msql_regcase (PHP 3, PHP 4 >= 4.0.0)

Make regular expression for case insensitive match

See sql_regcase().

msql_result (PHP 3, PHP 4 >= 4.0.0)

Get result data

int **msql_result** (int query_identifier, int i, mixed field) \linebreak

Returns the contents of the cell at the row and offset in the specified mSQL result set.

msql_result() returns the contents of one cell from a mSQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (fieldname.tablename). If the column name has been aliased ('select foo as bar from ...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **msql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: msql_fetch_row(), msql_fetch_array(), and msql_fetch_object().

msql_select_db (PHP 3, PHP 4 >= 4.0.0)

Select mSQL database

int **msql_select_db** (string database_name, int link_identifier) \linebreak

Returns TRUE on success, FALSE on error.

msql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if msql_connect() was called, and use it.

Every subsequent call to msql_query() will be made on the active database.

See also: msql_connect(), msql_pconnect(), and msql_query().

msql_selectdb (PHP 3, PHP 4 >= 4.0.0)

Select mSQL database

See msql_select_db().

msql_tablename (PHP 3, PHP 4 >= 4.0.0)

Get table name of field

string **msql_tablename** (int query_identifier, int field) \linebreak

msql_tablename() takes a result pointer returned by the msql_list_tables() function as well as an integer index and returns the name of a table. The msql_numrows() function may be used to determine the number of tables in the result pointer.

Esempio 1. msql_tablename() example

```
<?php
msql_connect ("localhost");
$result = msql_list_tables ("wisconsin");
$i = 0;
while ($i < msql_numrows ($result)) {
    $tb_names[$i] = msql_tablename ($result, $i);
    echo $tb_names[$i] . "<BR>";
    $i++;
}
?>
```

LXII. MySQL Functions

These functions allow you to access MySQL database servers. More information about MySQL can be found at http://www.mysql.com/.

Documentation for MySQL can be found at http://www.mysql.com/documentation/.

Requirements

In order to have these functions available, you must compile PHP with MySQL support.

Installation

By using the `--with-mysql` configuration option you enable PHP to access MySQL databases. If you use this option without specifying the path to MySQL, PHP will use the built-in MySQL client libraries. With PHP4 MySQL support is always enabled; if you don't specify the configure option, the bundled libraries are used. Users who run other applications that use MySQL (for example, running PHP 3 and PHP 4 as concurrent apache modules, or auth-mysql) should always specify the path to MySQL: `--with-mysql=/path/to/mysql`. This will force PHP to use the client libraries installed by MySQL, avoiding any conflicts.

Runtime Configuration

The behaviour of the MySQL functions is affected by settings in the global configuration file `php.ini`.

Tabella 1. MySQL Configuration Options

Name	Default	Changeable
mysql.allow_persistent	"On"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.default_port	NULL	PHP_INI_ALL
mysql.default_socket	NULL	PHP_INI_ALL
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL
mysql.default_password	NULL	PHP_INI_ALL

For further details and definition of the PHP_INI_* constants see ini_set().

Here is a short explanation of the configuration directives.

`mysql.allow_persistent` boolean

Wether to allow persistent connections to MySQL.

`mysql.max_persistent` integer

The maximum number of persistent MySQL connections per process.

`mysql.max_links` integer

The maximum number of MySQL connections per process, including persistent connections.

`mysql.default_port` string

The default TCP port number to use when connecting to the database server if no other port is specified. If no default is specified, the port will be obtained from the `MYSQL_TCP_PORT` environment variable, the `mysql-tcp` entry in `/etc/services` or the compile-time `MYSQL_PORT` constant, in that order. Win32 will only use the `MYSQL_PORT` constant.

`mysql.default_socket` string

The default socket name to use when connecting to a local database server if no other socket name is specified.

`mysql.default_host` string

The default server host to use when connecting to the database server if no other host is specified. Doesn't apply in safe mode.

`mysql.default_user` string

The default user name to use when connecting to the database server if no other name is specified. Doesn't apply in safe mode.

`mysql.default_password` string

The default password to use when connecting to the database server if no other password is specified. Doesn't apply in safe mode.

Resource types

There are two resource types used in the MySQL module. The first one is the link identifier for a database connection, the second a resource which helds the result of a query.

Predefined constants

The function mysql_fetch_array() uses a constant for the different types of result arrays. The

following constants are defined:

Tabella 2. MySQL fetch constants

constant	meaning
MYSQL_ASSOC	Columns are returned into the array having the fieldname as the array index.
MYSQL_BOTH	Columns are returned into the array having both a numerical index and the fieldname as the array index.
MYSQL_NUM	Columns are returned into the array having a numerical index to the fields. This index starts with 0, the first field in the result.
MYSQL_STORE_RESULT	Specifies that the MySQL result should be buffered.
MYSQL_USE_RESULT	Specifies that the MySQL result should not be buffered.

Examples

This simple example shows how to connect, execute a query, print resulting rows and disconnect from a MySQL database.

Esempio 1. MySQL extension overview example

```
<?php
    /* Connecting, selecting database */
    $link = mysql_connect("mysql_host", "mysql_user", "mysql_password")
        or die("Could not connect");
    print "Connected successfully";
    mysql_select_db("my_database") or die("Could not select database");

    /* Performing SQL query */
    $query = "SELECT * FROM my_table";
    $result = mysql_query($query) or die("Query failed");

    /* Printing results in HTML */
    print "<table>\n";
    while ($line = mysql_fetch_array($result, MYSQL_ASSOC)) {
        print "\t<tr>\n";
        foreach ($line as $col_value) {
            print "\t\t<td>$col_value</td>\n";
        }
        print "\t</tr>\n";
    }
    print "</table>\n";

    /* Free resultset */
```

```
    mysql_free_result($result);

    /* Closing connection */
    mysql_close($link);
?>
```

mysql_affected_rows (PHP 3, PHP 4 >= 4.0.0)

Get number of affected rows in previous MySQL operation

int **mysql_affected_rows** ([resource link_identifier]) \linebreak

mysql_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query associated with *link_identifier*. If the link identifier isn't specified, the last link opened by mysql_connect() is assumed.

> **Nota:** If you are using transactions, you need to call **mysql_affected_rows()** after your INSERT, UPDATE, or DELETE query, not after the commit.

If the last query was a DELETE query with no WHERE clause, all of the records will have been deleted from the table but this function will return zero.

> **Nota:** When using UPDATE, MySQL will not update columns where the new value is the same as the old value. This creates the possiblity that **mysql_affected_rows()** may not actually equal the number of rows matched, only the number of rows that were literally affected by the query.

mysql_affected_rows() does not work with SELECT statements; only on statements which modify records. To retrieve the number of rows returned by a SELECT, use mysql_num_rows().

If the last query failed, this function will return -1.

Esempio 1. Delete-Query

```
<?php
    /* connect to database */
    mysql_pconnect("localhost", "mysql_user", "mysql_password") or
        die ("Could not connect");

    /* this should return the correct numbers of deleted records */
    mysql_query("DELETE FROM mytable WHERE id < 10");
    printf ("Records deleted: %d\n", mysql_affected_rows());

    /* without a where clause in a delete statement, it should return 0 */
    mysql_query("DELETE FROM mytable");
    printf ("Records deleted: %d\n", mysql_affected_rows());
?>
```

The above example would produce the following output:

```
Records deleted: 10
Records deleted: 0
```

Esempio 2. Update-Query

```
<?php
    /* connect to database */
    mysql_pconnect("localhost", "mysql_user", "mysql_password") or
        die ("Could not connect");

    /* Update records */
    mysql_query("UPDATE mytable SET used=1 WHERE id < 10");
    printf ("Updated records: %d\n", mysql_affected_rows());
mysql_query("COMMIT");
?>
```

The above example would produce the following output:

```
Updated Records: 10
```

See also: mysql_num_rows(), mysql_info().

mysql_change_user (PHP 3>= 3.0.13)

Change logged in user of the active connection

int **mysql_change_user** (string user, string password [, string database [, resource link_identifier]]) \linebreak

mysql_change_user() changes the logged in user of the current active connection, or the connection given by the optional *link_identifier* parameter. If a database is specified, this will be the current database after the user has been changed. If the new user and password authorization fails, the current connected user stays active. Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

> **Nota:** This function was introduced in PHP 3.0.13 and requires MySQL 3.23.3 or higher. It is not available in PHP 4.

mysql_character_set_name (PHP 4 CVS only)

Returns the name of the character set

int **mysql_character_set_name** ([resource link_identifier]) \linebreak

mysql_character_set_name() returns the default character set name for the current connection.

Esempio 1. mysql_character_set_name() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$charset = mysql_character_set_name($link);
printf ("current character set is %s\n", $charset);
?>
```

The above example would produce the following output:

```
latin1
```

See also: mysql_real_escape_string()

mysql_close (PHP 3, PHP 4 >= 4.0.0)

Close MySQL connection

bool **mysql_close** ([resource link_identifier]) \linebreak

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

mysql_close() closes the connection to the MySQL server that's associated with the specified link identifier. If *link_identifier* isn't specified, the last opened link is used.

Using **mysql_close()** isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution. See also freeing resources.

> **Nota: mysql_close()** will not close persistent links created by mysql_pconnect().

Esempio 1. MySQL close example

```
<?php
    $link = mysql_connect("localhost", "mysql_user", "mysql_password")
        or exit("Could not connect");
    print ("Connected successfully");
    mysql_close($link);
?>
```

See also: mysql_connect(), and mysql_pconnect().

mysql_connect (PHP 3, PHP 4 >= 4.0.0)

Open a connection to a MySQL Server

resource **mysql_connect** ([string server [, string username [, string password [, bool new_link]]]]) \linebreak

Returns a MySQL link identifier on success, or FALSE on failure.

mysql_connect() establishes a connection to a MySQL server. The following defaults are assumed for missing optional parameters: *server* = 'localhost:3306', *username* = name of the user that owns the server process and *password* = empty password.

The *server* parameter can also include a port number. eg. "hostname:port" or a path to a socket eg. ":/path/to/socket" for the localhost.

> **Nota:** Support for ":port" was added in PHP 3.0B4.
>
> Support for ":/path/to/socket" was added in PHP 3.0.10.
>
> You can suppress the error message on failure by prepending a @ to the function name.

If a second call is made to **mysql_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned. The *new_link* parameter modifies this behavior and makes **mysql_connect()** always open a new link, even if **mysql_connect()** was called before with the same parameters.

> **Nota:** The *new_link* parameter became available in PHP 4.2.0

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling mysql_close().

Esempio 1. MySQL connect example

```
<?php
    $link = mysql_connect("localhost", "mysql_user", "mysql_password")
        or die("Could not connect");
    print ("Connected successfully");
    mysql_close($link);
?>
```

See also mysql_pconnect() and mysql_close().

mysql_create_db (PHP 3, PHP 4 >= 4.0.0)

Create a MySQL database

bool **mysql_create_db** (string database name [, resource link_identifier]) \linebreak

mysql_create_db() attempts to create a new database on the server associated with the specified link identifier.

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

Esempio 1. MySQL create database example

```
<?php
    $link = mysql_pconnect("localhost", "mysql_user", "mysql_password")
        or exit("Could not connect");

    if (mysql_create_db("my_db")) {
        print ("Database created successfully\n");
    } else {
        printf ("Error creating database: %s\n", mysql_error ());
    }
?>
```

For downwards compatibility **mysql_createdb()** can also be used. This is deprecated, however.

> **Nota:** The function **mysql_create_db()** is deprecated. It is preferable to use mysql_query() to issue a SQL CREATE DATABASE Statement instead.

See also: mysql_drop_db(), mysql_query().

mysql_data_seek (PHP 3, PHP 4 >= 4.0.0)

Move internal result pointer

bool **mysql_data_seek** (resource result_identifier, int row_number) \linebreak

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

mysql_data_seek() moves the internal row pointer of the MySQL result associated with the specified result identifier to point to the specified row number. The next call to mysql_fetch_row() would return that row.

Row_number starts at 0. The *row_number* should be a value in the range from 0 to mysql_num_rows - 1.

> **Nota:** The function **mysql_data_seek()** can be used in conjunction only with mysql_query(), not with mysql_unbuffered_query().

Esempio 1. MySQL data seek example

```
<?php
    $link = mysql_pconnect("localhost", "mysql_user", "mysql_password")
```

```
        or die("Could not connect");

    mysql_select_db("samp_db")
        or exit("Could not select database");

    $query = "SELECT last_name, first_name FROM friends";
    $result = mysql_query($query)
        or die("Query failed");

    /* fetch rows in reverse order */
    for ($i = mysql_num_rows($result) - 1; $i >= 0; $i--) {
        if (!mysql_data_seek($result, $i)) {
            echo "Cannot seek to row $i\n";
            continue;
        }

        if(!($row = mysql_fetch_object($result)))
            continue;

        echo "$row->last_name $row->first_name<br />\n";
    }

    mysql_free_result($result);
?>
```

See also: mysql_query(), mysql_num_rows().

mysql_db_name (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Get result data

string **mysql_db_name** (resource result, int row [, mixed field]) \linebreak

mysql_db_name() takes as its first parameter the result pointer from a call to mysql_list_dbs(). The *row* parameter is an index into the result set.

If an error occurs, FALSE is returned. Use mysql_errno() and mysql_error() to determine the nature of the error.

Esempio 1. mysql_db_name() example

```
<?php
    error_reporting(E_ALL);

    mysql_connect('dbhost', 'username', 'password');
    $db_list = mysql_list_dbs();

    $i = 0;
    $cnt = mysql_num_rows($db_list);
    while ($i < $cnt) {
        echo mysql_db_name($db_list, $i) . "\n";
        $i++;
```

```
    }
?>
```

For backward compatibility, **mysql_dbname()** is also accepted. This is deprecated, however.

mysql_db_query (PHP 3, PHP 4 >= 4.0.0)

Send a MySQL query

resource **mysql_db_query** (string database, string query [, resource link_identifier]) \linebreak

Returns a positive MySQL result resource to the query result, or FALSE on error.

mysql_db_query() selects a database and executes a query on it. If the optional link identifier isn't specified, the function will try to find an open link to the MySQL server and if no such link is found it'll try to create one as if mysql_connect() was called with no arguments

See also mysql_connect() and mysql_query().

> **Nota:** This function has been deprecated since PHP 4.0.6. Do not use this function. Use mysql_select_db() and mysql_query() instead.

mysql_drop_db (PHP 3, PHP 4 >= 4.0.0)

Drop (delete) a MySQL database

bool **mysql_drop_db** (string database_name [, resource link_identifier]) \linebreak

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

mysql_drop_db() attempts to drop (remove) an entire database from the server associated with the specified link identifier.

For downward compatibility **mysql_dropdb()** can also be used. This is deprecated, however.

> **Nota:** The function **mysql_drop_db()** is deprecated. It is prefarable to use mysql_query() to issue a SQL DROP DATABASE statement instead.

See also: mysql_create_db(), mysql_query().

mysql_errno (PHP 3, PHP 4 >= 4.0.0)

Returns the numerical value of the error message from previous MySQL operation

int **mysql_errno** ([resource link_identifier]) \linebreak

Returns the error number from the last MySQL function, or `0` (zero) if no error occurred.

Errors coming back from the MySQL database backend no longer issue warnings. Instead, use **mysql_errno()** to retrieve the error code. Note that this function only returns the error code from the most recently executed MySQL function (not including mysql_error() and **mysql_errno()**), so if you want to use it, make sure you check the value before calling another MySQL function.

Esempio 1. mysql_errno Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password");

    mysql_select_db("nonexistentdb");
    echo mysql_errno() . ": " . mysql_error(). "\n";

    mysql_select_db("kossu");
    mysql_query("SELECT * FROM nonexistenttable");
    echo mysql_errno() . ": " . mysql_error() . "\n";
?>
```

The above example would produce the following output:

```
1049: Unknown database ’nonexistentdb’
1146: Table ’kossu.nonexistenttable’ doesn’t exist
```

See also: mysql_error()

mysql_error (PHP 3, PHP 4 >= 4.0.0)

Returns the text of the error message from previous MySQL operation

string **mysql_error** ([resource link_identifier]) \linebreak

Returns the error text from the last MySQL function, or `”` (the empty string) if no error occurred.

Errors coming back from the MySQL database backend no longer issue warnings. Instead, use **mysql_error()** to retrieve the error text. Note that this function only returns the error text from the most recently executed MySQL function (not including **mysql_error()** and mysql_errno()), so if you want to use it, make sure you check the value before calling another MySQL function.

Esempio 1. mysql_error Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password");
```

```
    mysql_select_db("nonexistentdb");
    echo mysql_errno() . ": " . mysql_error(). "\n";

    mysql_select_db("kossu");
    mysql_query("SELECT * FROM nonexistenttable");
    echo mysql_errno() . ": " . mysql_error() . "\n";
?>
```

The above example would produce the following output:

```
1049: Unknown database 'nonexistentdb'
1146: Table 'kossu.nonexistenttable' doesn't exist
```

See also: mysql_errno()

mysql_escape_string (PHP 4 >= 4.0.3)

Escapes a string for use in a mysql_query.

string **mysql_escape_string** (string unescaped_string) \linebreak

This function will escape the *unescaped_string*, so that it is safe to place it in a mysql_query().

> **Nota: mysql_escape_string()** does not escape `%` and `_`.
>
> This function is identical to mysql_real_escape_string() except that mysql_real_escape_string() takes a connection handler and escapes the string according to the current character set. **mysql_escape_string()** does not take a connection argument and does not respect the current charset setting.

Esempio 1. mysql_real_escape_string() example

```
<?php
    $link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
    $item = "Zak's Laptop";
    $escaped_item = mysql_real_escape_string($item);
    printf ("Escaped string: %s\n", $escaped_item);
?>
```

The above example would produce the following output:

```
Escaped string: Zak\'s Laptop
```

See also mysql_real_escape_string(), addslashes(), and the magic_quotes_gpc directive.

mysql_fetch_array (PHP 3, PHP 4 >= 4.0.0)

Fetch a result row as an associative array, a numeric array, or both.

array **mysql_fetch_array** (resource result [, int result_type]) \linebreak

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

mysql_fetch_array() is an extended version of mysql_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you must use the numeric index of the column or make an alias for the column. For aliased columns, you cannot access the contents with the original column name (by using 'field' in this example).

Esempio 1. Query with duplicate field names

```
select table1.field as foo table2.field as bar from table1, table2
```

An important thing to note is that using **mysql_fetch_array()** is *not significantly* slower than using mysql_fetch_row(), while it provides a significant added value.

The optional second argument *result_type* in **mysql_fetch_array()** is a constant and can take the following values: MYSQL_ASSOC, MYSQL_NUM, and MYSQL_BOTH. This feature was added in PHP 3.0.7. MYSQL_BOTH is the default for this argument.

By using MYSQL_BOTH, you'll get an array with both associative and number indices. Using MYSQL_ASSOC, you only get associative indices (as mysql_fetch_assoc() works), using MYSQL_NUM, you only get number indices (as mysql_fetch_row() works).

Esempio 2. mysql_fetch_array with MYSQL_NUM

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    mysql_select_db("mydb");

    $result = mysql_query("SELECT id, name FROM mytable");

    while ($row = mysql_fetch_array($result, MYSQL_NUM)) {
        printf ("ID: %s  Name: %s", $row[0], $row[1]);
    }
```

```
    mysql_free_result($result);
?>
```

Esempio 3. mysql_fetch_array with MYSQL_ASSOC

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    mysql_select_db("mydb");

    $result = mysql_query("SELECT id, name FROM mytable");

    while ($row = mysql_fetch_array($result, MYSQL_ASSOC)) {
        printf ("ID: %s  Name: %s", $row["id"], $row["name"]);
    }

    mysql_free_result($result);
?>
```

Esempio 4. mysql_fetch_array with MYSQL_BOTH

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    mysql_select_db("mydb");

    $result = mysql_query("SELECT id, name FROM mytable");

    while ($row = mysql_fetch_array($result, MYSQL_BOTH)) {
        printf ("ID: %s  Name: %s", $row[0], $row["name"]);
    }

    mysql_free_result($result);
?>
```

For further details, see also mysql_fetch_row() and mysql_fetch_assoc().

mysql_fetch_assoc (PHP 4 >= 4.0.3)

Fetch a result row as an associative array

array **mysql_fetch_assoc** (resource result) \linebreak

Returns an associative array that corresponds to the fetched row, or `FALSE` if there are no more rows.

mysql_fetch_assoc() is equivalent to calling mysql_fetch_array() with MYSQL_ASSOC for the optional second parameter. It only returns an associative array. This is the way mysql_fetch_array() originally worked. If you need the numeric indices as well as the associative, use mysql_fetch_array().

If two or more columns of the result have the same field names, the last column will take precedence. To access the other column(s) of the same name, you either need to access the result with numeric indices by using mysql_fetch_row() or add alias names. See the example at the mysql_fetch_array() description about aliases.

An important thing to note is that using **mysql_fetch_assoc()** is *not significantly* slower than using mysql_fetch_row(), while it provides a significant added value.

Esempio 1. mysql_fetch_assoc()

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password");
    mysql_select_db("mydb");
    $query = "select * from table";
    $result = mysql_query($query);
    while ($row = mysql_fetch_assoc($result)) {
        echo $row["user_id"];
        echo $row["fullname"];
    }
    mysql_free_result($result);
?>
```

For further details, see also mysql_fetch_row(), mysql_fetch_array() and mysql_query().

mysql_fetch_field (PHP 3, PHP 4 >= 4.0.0)

Get column information from a result and return as an object

object **mysql_fetch_field** (resource result [, int field_offset]) \linebreak

Returns an object containing field information.

mysql_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retrieved by **mysql_fetch_field()** is retrieved.

The properties of the object are:

- name - column name
- table - name of the table the column belongs to
- max_length - maximum length of the column
- not_null - 1 if the column cannot be `NULL`
- primary_key - 1 if the column is a primary key
- unique_key - 1 if the column is a unique key

- multiple_key - 1 if the column is a non-unique key
- numeric - 1 if the column is numeric
- blob - 1 if the column is a BLOB
- type - the type of the column
- unsigned - 1 if the column is unsigned
- zerofill - 1 if the column is zero-filled

Esempio 1. mysql_fetch_field()

```
<?php
mysql_connect('localhost:3306', $user, $password)
    or die ("Could not connect");
mysql_select_db("database");
$result = mysql_query("select * from table")
    or die("Query failed");
/* get column metadata */
$i = 0;
while ($i < mysql_num_fields($result)) {
    echo "Information for column $i:<BR>\n";
    $meta = mysql_fetch_field($result);
    if (!$meta) {
        echo "No information available<BR>\n";
    }
    echo "<pre>
blob:         $meta->blob
max_length:   $meta->max_length
multiple_key: $meta->multiple_key
name:         $meta->name
not_null:     $meta->not_null
numeric:      $meta->numeric
primary_key:  $meta->primary_key
table:        $meta->table
type:         $meta->type
unique_key:   $meta->unique_key
unsigned:     $meta->unsigned
zerofill:     $meta->zerofill
</pre>";
    $i++;
}
mysql_free_result($result);
?>
```

See also mysql_field_seek().

mysql_fetch_lengths (PHP 3, PHP 4 >= 4.0.0)

Get the length of each output in a result

array **mysql_fetch_lengths** (resource result) \linebreak

Returns an array that corresponds to the lengths of each field in the last row fetched by mysql_fetch_row(), or FALSE on error.

mysql_fetch_lengths() stores the lengths of each result column in the last row returned by mysql_fetch_row(), mysql_fetch_array(), and mysql_fetch_object() in an array, starting at offset 0.

See also: mysql_fetch_row().

mysql_fetch_object (PHP 3, PHP 4 >= 4.0.0)

Fetch a result row as an object

object **mysql_fetch_object** (resource result) \linebreak

Returns an object with properties that correspond to the fetched row, or FALSE if there are no more rows.

mysql_fetch_object() is similar to mysql_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

```
<?php

/* this is valid */
echo $row->field;
/* this is invalid */
echo $row->0;

?>
```

Speed-wise, the function is identical to mysql_fetch_array(), and almost as quick as mysql_fetch_row() (the difference is insignificant).

Esempio 1. mysql_fetch_object() example

```
<?php
mysql_connect("hostname", "user", "password");
mysql_select_db($db);
$result = mysql_query("select * from table");
while ($row = mysql_fetch_object($result)) {
    echo $row->user_id;
    echo $row->fullname;
}
```

```
mysql_free_result($result);
?>
```

See also: mysql_fetch_array() and mysql_fetch_row().

mysql_fetch_row (PHP 3, PHP 4 >= 4.0.0)

Get a result row as an enumerated array

array **mysql_fetch_row** (resource result) \linebreak

Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

mysql_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **mysql_fetch_row()** would return the next row in the result set, or `FALSE` if there are no more rows.

See also: mysql_fetch_array(), mysql_fetch_object(), mysql_data_seek(), mysql_fetch_lengths(), and mysql_result().

mysql_field_flags (PHP 3, PHP 4 >= 4.0.0)

Get the flags associated with the specified field in a result

string **mysql_field_flags** (resource result, int field_offset) \linebreak

mysql_field_flags() returns the field flags of the specified field. The flags are reported as a single word per flag separated by a single space, so that you can split the returned value using explode().

The following flags are reported, if your version of MySQL is current enough to support them: "not_null", "primary_key", "unique_key", "multiple_key", "blob", "unsigned", "zerofill", "binary", "enum", "auto_increment", "timestamp".

For downward compatibility **mysql_fieldflags()** can also be used. This is deprecated, however.

mysql_field_len (PHP 3, PHP 4 >= 4.0.0)

Returns the length of the specified field

int **mysql_field_len** (resource result, int field_offset) \linebreak

mysql_field_len() returns the length of the specified field.

For downward compatibility **mysql_fieldlen()** can also be used. This is deprecated, however.

mysql_field_name (PHP 3, PHP 4 >= 4.0.0)

Get the name of the specified field in a result

string **mysql_field_name** (resource result, int field_index) \linebreak

mysql_field_name() returns the name of the specified field index. *result* must be a valid result identifier and *field_index* is the numerical offset of the field.

> **Nota:** *field_index* starts at 0.
>
> e.g. The index of the third field would actually be 2, the index of the fourth field would be 3 and so on.

Esempio 1. mysql_field_name() example

```
/* The users table consists of three fields:
 *   user_id
 *   username
 *   password.
 */
$link = mysql_connect('localhost', "mysql_user", "mysql_password");
mysql_select_db($dbname, $link)
    or die("Could not set $dbname");
$res = mysql_query("select * from users", $link);

echo mysql_field_name($res, 0) . "\n";
echo mysql_field_name($res, 2);
```

The above example would produce the following output:

```
user_id
password
```

For downwards compatibility **mysql_fieldname()** can also be used. This is deprecated, however.

mysql_field_seek (PHP 3, PHP 4 >= 4.0.0)

Set result pointer to a specified field offset

int **mysql_field_seek** (resource result, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to mysql_fetch_field() doesn't include a field offset, the field offset specified in **mysql_field_seek()** will be returned.

See also: mysql_fetch_field().

mysql_field_table (PHP 3, PHP 4 >= 4.0.0)

Get name of the table the specified field is in

string **mysql_field_table** (resource result, int field_offset) \linebreak

Returns the name of the table that the specifed field is in.

For downward compatibility **mysql_fieldtable()** can also be used. This is deprecated, however.

mysql_field_type (PHP 3, PHP 4 >= 4.0.0)

Get the type of the specified field in a result

string **mysql_field_type** (iresource result, int field_offset) \linebreak

mysql_field_type() is similar to the mysql_field_name() function. The arguments are identical, but the field type is returned instead. The field type will be one of "int", "real", "string", "blob", and others as detailed in the MySQL documentation (http://www.mysql.com/documentation/).

Esempio 1. MySQL field types

```
<?php
    mysql_connect("localhost", "mysql_username", "mysql_password");
    mysql_select_db("mysql");
    $result = mysql_query("SELECT * FROM func");
    $fields = mysql_num_fields($result);
    $rows   = mysql_num_rows($result);
    $table = mysql_field_table($result, 0);
    echo "Your '".$table."' table has ".$fields." fields and ".$rows." record(s)\n";
    echo "The table has the following fields:\n";
    for ($i=0; $i < $fields; $i++) {
        $type  = mysql_field_type($result, $i);
        $name  = mysql_field_name($result, $i);
        $len   = mysql_field_len($result, $i);
        $flags = mysql_field_flags($result, $i);
        echo $type." ".$name." ".$len." ".$flags."\n";
    }
    mysql_free_result($result);
    mysql_close();
?>
```

The above example would produce the following output:

```
Your 'func' table has 4 fields and 1 record(s)
The table has the following fields:
string name 64 not_null primary_key binary
int ret 1 not_null
```

```
string dl 128 not_null
string type 9 not_null enum
```

For downward compatibility **mysql_fieldtype()** can also be used. This is deprecated, however.

mysql_free_result (PHP 3, PHP 4 >= 4.0.0)

Free result memory

bool **mysql_free_result** (resource result) \linebreak

mysql_free_result() will free all memory associated with the result identifier *result*.

mysql_free_result() only needs to be called if you are concerned about how much memory is being used for queries that return large result sets. All associated result memory is automatically freed at the end of the script's execution.

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

For downward compatibility **mysql_freeresult()** can also be used. This is deprecated, however.

mysql_get_client_info (PHP 4 >= 4.0.5)

Get MySQL client info

string **mysql_get_client_info** (void) \linebreak

mysql_get_client_info() returns a string that represents the client library version.

Esempio 1. mysql_get_client_info Example

```
<?php
    printf ("MySQL client info: %s\n", mysql_get_client_info());
?>
```

The above example would produce the following output:

```
MySQL client info: 3.23.39
```

See also: mysql_get_host_info(), mysql_get_proto_info() and mysql_get_server_info().

mysql_get_host_info (PHP 4 >= 4.0.5)

Get MySQL host info

string **mysql_get_host_info** ([resource link_identifier]) \linebreak

mysql_get_host_info() returns a string describing the type of connection in use for the connection *link_identifier*, including the server host name. If *link_identifier* is omitted, the last opened connection will be used.

Esempio 1. mysql_get_host_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    printf ("MySQL host info: %s\n", mysql_get_host_info());
?>
```

The above example would produce the following output:

```
MySQL host info: Localhost via UNIX socket
```

See also: mysql_get_client_info(), mysql_get_proto_info() and mysql_get_server_info().

mysql_get_proto_info (PHP 4 >= 4.0.5)

Get MySQL protocol info

int **mysql_get_proto_info** ([resource link_identifier]) \linebreak

mysql_get_proto_info() returns the protocol version used by connection *link_identifier*. If *link_identifier* is omitted, the last opened connection will be used.

Esempio 1. mysql_get_proto_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    printf ("MySQL protocol version: %s\n", mysql_get_proto_info());
?>
```

The above example would produce the following output:

```
MySQL protocol version : 10
```

See also: mysql_get_client_info(), mysql_get_host_info() and mysql_get_server_info().

mysql_get_server_info (PHP 4 >= 4.0.5)

Get MySQL server info

string **mysql_get_server_info** ([resource link_identifier]) \linebreak

mysql_get_server_info() returns the server version used by connection `link_identifier`. If `link_identifier` is omitted, the last opened connection will be used.

Esempio 1. mysql_get_server_info Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    printf ("MySQL server version: %s\n", mysql_get_server_info());
?>
```

The above example would produce the following output:

```
MySQL server version: 4.0.1-alpha
```

See also: mysql_get_client_info(), mysql_get_host_info() and mysql_get_proto_info().

mysql_info (PHP 4 CVS only)

Get information about the most recent query

string **mysql_info** ([resource link_identifier]) \linebreak

mysql_info() returns detailed information about the last query using the given `link_identifier`. If `link_identifier` isn't specified, the last opened link is assumed.

mysql_info() returns a string for all statements listed below. For all other `FALSE`. The string format depends on the given statement.

Esempio 1. Relevant MySQL Statements

```
INSERT INTO ... SELECT ...
String format: Records: 23 Duplicates: 0 Warnings: 0
INSERT INTO ... VALUES (...),(...),(...)...
String format: Records: 37 Duplicates: 0 Warnings: 0
LOAD DATA INFILE ...
String format: Records: 42 Deleted: 0 Skipped: 0 Warnings: 0
ALTER TABLE
```

```
String format: Records: 60 Duplicates: 0 Warnings: 0
UPDATE
String format: Rows matched: 65 Changed: 65 Warnings: 0
```

The numbers are only for illustrating purpose; their values will correspond to the query.

> **Nota: mysql_info()** returns a non-`FALSE` value for the INSERT ... VALUES statement only if multiple value lists are specified in the statement.

See also: mysql_affected_rows()

mysql_insert_id (PHP 3, PHP 4 >= 4.0.0)

Get the id generated from the previous INSERT operation

int **mysql_insert_id** ([resource link_identifier]) \linebreak

mysql_insert_id() returns the ID generated for an AUTO_INCREMENT column by the previous INSERT query using the given *`link_identifier`*. If *`link_identifier`* isn't specified, the last opened link is assumed.

mysql_insert_id() returns 0 if the previous query does not generate an AUTO_INCREMENT value. If you need to save the value for later, be sure to call **mysql_insert_id()** immediately after the query that generates the value.

> **Nota:** The value of the MySQL SQL function `LAST_INSERT_ID()` always contains the most recently generated AUTO_INCREMENT value, and is not reset between queries.

Attenzione

mysql_insert_id() converts the return type of the native MySQL C API function `mysql_insert_id()` to a type of `long` (named int in PHP). If your AUTO_INCREMENT column has a column type of BIGINT, the value returned by **mysql_insert_id()** will be incorrect. Instead, use the internal MySQL SQL function `LAST_INSERT_ID()` in an SQL query.

Esempio 1. mysql_insert_id Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password") or
        die("could not connect");
    mysql_select_db("mydb");

    mysql_query("INSERT INTO mytable (product) values ('kossu')");
    printf ("Last inserted record has id %d\n", mysql_insert_id());
?>
```

See also: mysql_query().

mysql_list_dbs (PHP 3, PHP 4 >= 4.0.0)

List databases available on a MySQL server

resource **mysql_list_dbs** ([resource link_identifier]) \linebreak

mysql_list_dbs() will return a result pointer containing the databases available from the current mysql daemon. Use the mysql_tablename() function to traverse this result pointer, or any function for result tables.

Esempio 1. mysql_list_dbs() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$db_list = mysql_list_dbs($link);

while ($row = mysql_fetch_object($db_list)) {
    echo $row->Database . "\n";
}
?>
```

The above example would produce the following output:

```
database1
database2
database3
...
```

> **Nota:** The above code would just as easily work with mysql_fetch_row() or other similar functions.

For downward compatibility **mysql_listdbs()** can also be used. This is deprecated however.

See also mysql_db_name().

mysql_list_fields (PHP 3, PHP 4 >= 4.0.0)

List MySQL result fields

resource **mysql_list_fields** (string database_name, string table_name [, resource link_identifier]) \linebreak

mysql_list_fields() retrieves information about the given tablename. Arguments are the database name and the table name. A result pointer is returned which can be used with mysql_field_flags(), mysql_field_len(), mysql_field_name(), and mysql_field_type().

Esempio 1. mysql_list_fields() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');

$fields = mysql_list_fields("database1", "table1", $link);
$columns = mysql_num_fields($fields);

for ($i = 0; $i < $columns; $i++) {
    echo mysql_field_name($fields, $i) . "\n";
}
```

The above example would produce the following output:

```
field1
field2
field3
...
```

For downward compatibility **mysql_listfields()** can also be used. This is deprecated however.

mysql_list_processes (PHP 4 CVS only)

List MySQL processes

resource **mysql_list_processes** ([resource link_identifier]) \linebreak

mysql_list_processes() returns a result pointer describing the current server threads.

Esempio 1. mysql_list_processes() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
```

```
$result = mysql_list_processes($link);
while ($row = mysql_fetch_row($result)){
    printf("%s %s %s %s %s\n", $row["Id"], $row["Host"], $row["db"],
       $row["Command"], $row["Time"]);
}
mysql_free_result ($result);
?>
```

The above example would produce the following output:

```
1 localhost test Processlist 0
4 localhost mysql sleep 5
```

See also: mysql_thread_id()

mysql_list_tables (PHP 3, PHP 4 >= 4.0.0)

List tables in a MySQL database

resource **mysql_list_tables** (string database [, resource link_identifier]) \linebreak

mysql_list_tables() takes a database name and returns a result pointer much like the mysql_query() function. You can use the mysql_tablename() function to extract the actual table names from the result pointer, or any other result table function such as mysql_fetch_assoc().

The *database* parameter is the name of the database to retrieve the list of tables from. Upon failure, **mysql_list_tables()** returns FALSE.

For downward compatibility, the function alias named **mysql_listtables()** can be used. This is deprecated however and is not recommended.

Esempio 1. mysql_list_tables Example

```
<?php
    $dbname = 'mysql_dbname';

    if (!mysql_connect('mysql_host', 'mysql_user', 'mysql_password')) {
        print 'Could not connect to mysql';
        exit;
    }

    $result = mysql_list_tables($dbname);

    if (!$result) {
        print "DB Error, could not list tables\n";
        print 'MySQL Error: ' . mysql_error();
        exit;
```

```
    }

    while ($row = mysql_fetch_row($result)) {
        print "Table: $row[0]\n";
    }

    mysql_free_result($result);
?>
```

See also: mysql_list_dbs(), and mysql_tablename().

mysql_num_fields (PHP 3, PHP 4 >= 4.0.0)

Get number of fields in result

int **mysql_num_fields** (resource result) \linebreak

mysql_num_fields() returns the number of fields in a result set.

See also: mysql_db_query(), mysql_query(), mysql_fetch_field(), mysql_num_rows().

For downward compatibility **mysql_numfields()** can also be used. This is deprecated however.

mysql_num_rows (PHP 3, PHP 4 >= 4.0.0)

Get number of rows in result

int **mysql_num_rows** (resource result) \linebreak

mysql_num_rows() returns the number of rows in a result set. This command is only valid for SELECT statements. To retrieve the number of rows affected by a INSERT, UPDATE or DELETE query, use mysql_affected_rows().

Esempio 1. mysql_num_rows() example

```
<?php

$link = mysql_connect("localhost", "mysql_user", "mysql_password");
mysql_select_db("database", $link);

$result = mysql_query("SELECT * FROM table1", $link);
$num_rows = mysql_num_rows($result);

echo "$num_rows Rows\n";

?>
```

> **Nota:** If you use mysql_unbuffered_query(), **mysql_num_rows()** will not return the correct value until all the rows in the result set have been retrieved.

See also: mysql_affected_rows(), mysql_connect(), mysql_data_seek(), mysql_select_db(), and mysql_query().

For downward compatibility **mysql_numrows()** can also be used. This is deprecated however.

mysql_pconnect (PHP 3, PHP 4 >= 4.0.0)

Open a persistent connection to a MySQL server

resource **mysql_pconnect** ([string server [, string username [, string password]]]) \linebreak

Returns a positive MySQL persistent link identifier on success, or `FALSE` on error.

mysql_pconnect() establishes a connection to a MySQL server. The following defaults are assumed for missing optional parameters: *`server`* = 'localhost:3306', *`username`* = name of the user that owns the server process and *`password`* = empty password.

The *`server`* parameter can also include a port number. eg. "hostname:port" or a path to a socket eg. ":/path/to/socket" for the localhost.

> **Nota:** Support for ":port" was added in 3.0B4.
>
> Support for the ":/path/to/socket" was added in 3.0.10.

mysql_pconnect() acts very much like mysql_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (mysql_close() will not close links established by **mysql_pconnect()**).

This type of link is therefore called 'persistent'.

> **Nota:** Note, that these kind of links only work if you are using a module version of PHP. See the Persistent Database Connections section for more information.

> **Attenzione**
>
> Using persistent connections can require a bit of tuning of your Apache and MySQL configurations to ensure that you do not exceed the number of connections allowed by MySQL.

mysql_ping (PHP 4 CVS only)

Ping a server connection or reconnect if there is no connection

bool **mysql_ping** ([resource link_identifier]) \linebreak

mysql_ping() checks whether or not the connection to the server is working. If it has gone down, an automatic reconnection is attempted. This function can be used by scripts that remain idle for a long while, to check whether or not the server has closed the connection and reconnect if necessary. **mysql_ping()** returns `TRUE` if the connection to the server is working, otherwise `FALSE`.

See also: mysql_thread_id(), mysql_list_processes().

mysql_query (PHP 3, PHP 4 >= 4.0.0)

Send a MySQL query

resource **mysql_query** (string query [, resource link_identifier [, int result_mode]]) \linebreak

mysql_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If *`link_identifier`* isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if mysql_connect() was called with no arguments, and use it.

The optional *`result_mode`* parameter can be MYSQL_USE_RESULT and MYSQL_STORE_RESULT. It defaults to MYSQL_STORE_RESULT, so the result is buffered. See also mysql_unbuffered_query() for the counterpart of this behaviour.

> **Nota:** The query string should not end with a semicolon.

Only for SELECT,SHOW,EXPLAIN or DESCRIBE statements **mysql_query()** returns a resource identifier or `FALSE` if the query was not executed correctly. For other type of SQL statements, **mysql_query()** returns `TRUE` on success and `FALSE` on error. A non-`FALSE` return value means that the query was legal and could be executed by the server. It does not indicate anything about the number of rows affected or returned. It is perfectly possible for a query to succeed but affect no rows or return no rows.

The following query is syntactically invalid, so **mysql_query()** fails and returns `FALSE`:

Esempio 1. mysql_query()

```
<php
$result = mysql_query("SELECT * WHERE 1=1")
    or die("Invalid query");
?>
```

The following query is semantically invalid if `my_col` is not a column in the table `my_tbl`, so **mysql_query()** fails and returns `FALSE`:

Esempio 2. mysql_query()

```
<?php
$result = mysql_query("SELECT my_col FROM my_tbl")
    or die ("Invalid query");
?>
```

mysql_query() will also fail and return `FALSE` if you don't have permission to access the table(s) referenced by the query.

Assuming the query succeeds, you can call mysql_num_rows() to find out how many rows were returned for a SELECT statment or mysql_affected_rows() to find out how many rows were affected by a DELETE, INSERT, REPLACE, or UPDATE statement.

Only for SELECT,SHOW,DESCRIBE or EXPLAIN statements, **mysql_query()** returns a new result identifier that you can pass to mysql_fetch_array() and other functions dealing with result tables. When you are done with the result set, you can free the resources associated with it by calling mysql_free_result(). Although, the memory will automatically be freed at the end of the script's execution.

See also: mysql_num_rows(), mysql_affected_rows(), mysql_unbuffered_query(), mysql_free_result(), mysql_fetch_array(), mysql_fetch_row(), mysql_fetch_assoc(), mysql_result(), mysql_select_db(), and mysql_connect().

mysql_real_escape_string (PHP 4 CVS only)

Escapes special characters in a string for use in a SQL statement, taking into account the current charset of the connection.

string **mysql_real_escape_string** (string unescaped_string [, resource link_identifier]) \linebreak

This function will escape special characters in the *`unescaped_string`*, taking into account the current charset of the connection so that it is safe to place it in a mysql_query().

> **Nota: mysql_real_escape_string()** does not escape `%` and `_`.

Esempio 1. mysql_real_escape_string() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$item = "Zak's and Derick's Laptop";
$escaped_item = mysql_real_escape_string($item);
printf ("Escaped string: %s\n", $escaped_item);
?>
```

The above example would produce the following output:

```
Escaped string: Zak\'s and Derick\'s Laptop
```

See also: mysql_escape_string(), **mysql_character_set_name()**.

mysql_result (PHP 3, PHP 4 >= 4.0.0)

Get result data

mixed **mysql_result** (resource result, int row [, mixed field]) \linebreak

mysql_result() returns the contents of one cell from a MySQL result set. The field argument can be the field's offset, or the field's name, or the field's table dot field name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than **mysql_result()**. Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Calls to **mysql_result()** should not be mixed with calls to other functions that deal with the result set.

Recommended high-performance alternatives: mysql_fetch_row(), mysql_fetch_array(), and mysql_fetch_object().

mysql_select_db (PHP 3, PHP 4 >= 4.0.0)

Select a MySQL database

bool **mysql_select_db** (string database_name [, resource link_identifier]) \linebreak

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

mysql_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if mysql_connect() was called without arguments, and use it.

Every subsequent call to mysql_query() will be made on the active database.

See also: mysql_connect(), mysql_pconnect(), and mysql_query().

For downward compatibility **mysql_selectdb()** can also be used. This is deprecated however.

mysql_stat (PHP 4 CVS only)

Get current system status

string **mysql_stat** ([resource link_identifier]) \linebreak

mysql_stat() returns the current server status.

> **Nota: mysql_stat()** currently only returns status for uptime, threads, queries, open tables, flush tables and queries per second. For a complete list of other status variables you have to use the SHOW STATUS sql command.

Esempio 1. mysql_stat() example

```
<?php
$link = mysql_connect('localhost', "mysql_user", "mysql_password");
printf("%s\n", mysql_stat($link));
?>
```

The above example would produce the following output:

```
Uptime: 5380  Threads: 1  Questions: 1321299  Slow queries: 1  Opens: 26 Flush ta-
bles: 1  Open tables: 17  Queries per second avg: 245.595
```

mysql_tablename (PHP 3, PHP 4 >= 4.0.0)

Get table name of field

string **mysql_tablename** (resource result, int i) \linebreak

mysql_tablename() takes a result pointer returned by the mysql_list_tables() function as well as an integer index and returns the name of a table. The mysql_num_rows() function may be used to determine the number of tables in the result pointer.

Esempio 1. mysql_tablename() Example

```
<?php
    mysql_connect("localhost", "mysql_user", "mysql_password");
    $result = mysql_list_tables("mydb");

    for ($i = 0; $i < mysql_num_rows($result); $i++)
        printf ("Table: %s\n", mysql_tablename($result, $i));

    mysql_free_result($result);
?>
```

See also: mysql_list_tables().

mysql_thread_id (PHP 4 CVS only)

Return the current thread id

int **mysql_thread_id** ([resource link_identifier]) \linebreak

mysql_thread_id() returns the current thread id. If the connection is lost and you reconnect with mysql_ping(), the thread ID will change. This means you should not get the thread ID and store it for later. You should get it when you need it.

Esempio 1. mysql_list_processes() example

```
<?php
$link = mysql_connect('localhost', 'mysql_user', 'mysql_password');
$thread_id = mysql_thread_id($link);
if ($thread_id){
    printf ("current thread id is %d\n", $thread_id);
}
?>
```

The above example would produce the following output:

```
current thread id is 73
```

See also: mysql_ping(), mysql_list_processes().

mysql_unbuffered_query (PHP 4 >= 4.0.6)

Send an SQL query to MySQL, without fetching and buffering the result rows

resource **mysql_unbuffered_query** (string query [, resource link_identifier [, int result_mode]]) \linebreak

mysql_unbuffered_query() sends a SQL query *query* to MySQL, without fetching and buffering the result rows automatically, as mysql_query() does. On the one hand, this saves a considerable amount of memory with SQL queries that produce large result sets. On the other hand, you can start working on the result set immediately after the first row has been retrieved: you don't have to wait until the complete SQL query has been performed. When using multiple DB-connects, you have to specify the optional parameter *link_identifier*.

The optional *`result_mode`* parameter can be MYSQL_USE_RESULT and MYSQL_STORE_RESULT. It defaults to MYSQL_USE_RESULT, so the result is not buffered. See also mysql_query() for the counterpart of this behaviour.

> **Nota:** The benefits of **mysql_unbuffered_query()** come at a cost: You cannot use mysql_num_rows() on a result set returned from **mysql_unbuffered_query()**. You also have to fetch all result rows from an unbuffered SQL query, before you can send a new SQL query to MySQL.

See also: mysql_query().

LXIII. Mohawk Software session handler functions

msession is an interface to a high speed session daemon which can run either locally or remotely. It is designed to provide consistent session management for a PHP web farm.

The session server software can be found at http://www.mohawksoft.com/phoenix/.

msession_connect (PHP 4 >= 4.2.0)

Connect to msession server

bool **msession_connect** (string host, string port) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_count (PHP 4 >= 4.2.0)

Get session count

int **msession_count** (void) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_create (PHP 4 >= 4.2.0)

Create a session

bool **msession_create** (string session) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_destroy (PHP 4 >= 4.2.0)

Destroy a session

bool **msession_destroy** (string name) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_disconnect (PHP 4 >= 4.2.0)

Close connection to msession server

void **msession_disconnect** (void) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_find (PHP 4 >= 4.2.0)

Find value

array **msession_find** (string name, string value) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_get (PHP 4 >= 4.2.0)

Get value from session

string **msession_get** (string session, string name, string value) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_get_array (PHP 4 >= 4.2.0)

Get array of ... ?

array **msession_get_array** (string session) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_getdata (unknown)

Get data ... ?

string **msession_getdata** (string session) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_inc (PHP 4 >= 4.2.0)

Increment value in session

string **msession_inc** (string session, string name) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_list (PHP 4 >= 4.2.0)

List ... ?

array **msession_list** (void) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_listvar (PHP 4 >= 4.2.0)

List sessions with variable

array **msession_listvar** (string name) \linebreak

Returns an associative array of value, session for all sessions with a variable named `name`.

Used for searching sessions with common attributes.

msession_lock (PHP 4 >= 4.2.0)

Lock a session

int **msession_lock** (string name) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_plugin (PHP 4 >= 4.2.0)

Call an escape function within the msession personality plugin

string **msession_plugin** (string session, string val [, string param]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_randstr (PHP 4 >= 4.2.0)

Get random string

string **msession_randstr** (int param) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_set (PHP 4 >= 4.2.0)

Set value in session

bool **msession_set** (string session, string name, string value) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_set_array (PHP 4 >= 4.2.0)

Set array of ...

bool **msession_set_array** (string session, array tuples) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_setdata (unknown)

Set data ... ?

bool **msession_setdata** (string session, string value) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_timeout (PHP 4 >= 4.2.0)

Set/get session timeout

int **msession_timeout** (string session [, int param]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_uniq (PHP 4 >= 4.2.0)

Get uniq id

string **msession_uniq** (int param) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msession_unlock (PHP 4 >= 4.2.0)

Unlock a session

int **msession_unlock** (string session, int key) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

LXIV. muscat functions

muscat_close (4.0.5 - 4.2.0 only)

Shuts down the muscat session and releases any memory back to php. [Not back to the system, note!]

int **muscat_close** (resource muscat_handle) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

muscat_get (4.0.5 - 4.2.0 only)

Gets a line back from the core muscat api. Returns a literal FALSE when there is no more to get (as opposed to ""). Use === FALSE or !== FALSE to check for this

string **muscat_get** (resource muscat_handle) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

muscat_give (4.0.5 - 4.2.0 only)

Sends string to the core muscat api

int **muscat_give** (resource muscat_handle, string string) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

muscat_setup (4.0.5 - 4.2.0 only)

Creates a new muscat session and returns the handle. Size is the ammount of memory in bytes to allocate for muscat muscat_dir is the muscat installation dir e.g. "/usr/local/empower", it defaults to the compile time muscat directory

resource **muscat_setup** (int size [, string muscat_dir]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

muscat_setup_net (4.0.5 - 4.2.0 only)

Creates a new muscat session and returns the handle. muscat_host is the hostname to connect to port is the port number to connect to - actually takes exactly the same args as fsockopen

resource **muscat_setup_net** (string muscat_host, int port) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

LXV. Funzioni di rete

checkdnsrr (PHP 3, PHP 4 >= 4.0.0)

Controlla i record DNS relativi ad un host Internet o indirizzo IP

int **checkdnsrr** (string host [, string type]) \linebreak

Cerca i record DNS del tipo *`type`* corrispondenti a *`host`*. Restituisce vero se dei records sono trovati; falso se nessun record viene trovato o in caso di errore.

`type` può essere uno dei seguenti: A, MX, NS, SOA, PTR, CNAME, oppure ANY. Il default è MX.

`Host` può essere sia l'indirizzo IP in notazione decimale o il nome dell'host.

> **Nota:** Questa funzione non è implementata su piattaforme Windows

Vedere anche getmxrr(), gethostbyaddr(), gethostbyname(), gethostbynamel() e la man page named(8).

closelog (PHP 3, PHP 4 >= 4.0.0)

Chiude la connessione al logger di sistema

int **closelog** (void) \linebreak

closelog() chiude il descrittore usato per scrivere al logger di sistema. L'uso di **closelog()** è facoltativo.

Vedere anche define_syslog_variables(), syslog() e openlog().

debugger_off (PHP 3)

Disattiva il debugger interno PHP

int **debugger_off** (void) \linebreak

Disattiva il debugger interno PHP. Il debugger è ancora in fase di sviluppo.

debugger_on (PHP 3)

Attiva il debugger interno PHP

int **debugger_on** (string indirizzo) \linebreak

Attiva il debugger interno PHP, connettendolo ad *`indirizzo`*. Il debugger è in fase di sviluppo.

define_syslog_variables (PHP 3, PHP 4 >= 4.0.0)

Inizializza tutte le costanti collegate al syslog

void **define_syslog_variables** (void) \linebreak

Inizializza tutte le costanti usate nelle funzioni del syslog.

Vedere anche openlog(), syslog() e closelog().

fsockopen (PHP 3, PHP 4 >= 4.0.0)

Apre una connessione a un socket appartenente a un dominio Internet o Unix

int **fsockopen** (string hostname, int porta [, int errno [, string errstr [, float timeout]]]) \linebreak

Inizializza una connessione nel dominio Internet (AF_INET, usando TCP o UDP) o Unix (AF_UNIX). Per il dominio Internet, apre una connessione a un socket TCP verso l' *hostname* sulla porta *port*. *hostname* può essere in questo caso, sia un fully qualified domain name o un indirizzo IP. Per le connessioni UDP, è necessario specificare esplicitamente il protocollo, usando: '`udp://`' come prefisso di *hostname*. Per il dominio Unix, *hostname* viene utilizzato come percorso verso il socket, in questo caso, *porta* deve essere impostato a 0. Il parametro opzionale *timeout* può essere usato per impostare un timeout in secondi per la chiamata di sistema connect.

A partire da PHP 4.3.0, se si è compilato con il supporto OpenSSL, si può prefissare *hostname* con '`ssl://`' oppure '`tls://`' per utilizzare una connessione client SSL o TLS su una connessione TCP/IP per connettersi all'host remoto.

fsockopen() restituisce un puntatore a file che può essere usato nelle altre funzioni orientate ai file (come fgets(), fgetss(), fputs(), fclose() e feof()).

Se la chiamata non ha successo, viene restituito `FALSE` e se gli argomenti opzionali *errno* e *errstr* sono presenti, vengono impostati a indicare l'errore a livello di sistema che è avvenuto nella chiamata alla funzione `connect()` del sistema operativo. Se il valore di *errno* restituito è `0` e la funzione restituisce `FALSE`, è un'indicazione che l'errore è avvenuto prima della chiamata di `connect()`. Questo è molto probabilmente legato ad un problema di inizializzazione del socket. Si noti che gli argomenti *errno* e *errstr* verranno sempre passati by reference.

A seconda dell'ambiente operativo, il dominio Unix o l'opzionale timeout della connect potrebbero non essere disponibili.

Il socket viene aperto di default in modo blocking. Si può passare al modo non-blocking usando socket_set_blocking().

Esempio 1. Esempio di fsockopen()

```
<?php
$fp = fsockopen ("www.php.net", 80, $errno, $errstr, 30);
if (!$fp) {
    echo "$errstr ($errno)<br>\n";
} else {
    fputs ($fp, "GET / HTTP/1.0\r\nHost: www.php.net\r\n\r\n");
    while (!feof($fp)) {
        echo fgets ($fp,128);
```

```
    }
    fclose ($fp);
}
?>
```

L'esempio seguente mostra come ottenere data e ora tramite il servizio UDP "daytime" (porta 13) della vostra stessa macchina.

Esempio 2. Uso di connessione UDP

```
<?php
$fp = fsockopen("udp://127.0.0.1", 13, $errno, $errstr);
if (!$fp) {
    echo "ERRORE: $errno - $errstr<br>\n";
} else {
    fwrite($fp,"\n");
    echo fread($fp, 26);
    fclose($fp);
}
?>
```

> **Nota:** Il parametro timeout è stato introdotto nel PHP 3.0.9 e il supporto UDP è stato aggiunto nel PHP 4.

Vedere anche pfsockopen(), socket_set_blocking(), socket_set_timeout(), fgets(), fgetss(), fputs(), fclose(), feof() e l'estensione Curl.

gethostbyaddr (PHP 3, PHP 4 >= 4.0.0)

Ottiene l'host Internet corrispondente a un dato indirizzo IP

string **gethostbyaddr** (string indirizzo_ip) \linebreak

Restituisce l'hostname dell'host Internet specificato da *indirizzo_ip*. Se occorre un errore, restituisce *indirizzo_ip*.

Vedere anche gethostbyname().

gethostbyname (PHP 3, PHP 4 >= 4.0.0)

Ottiene l'indirizzo IP corrispondente a un dato hostname Internet

string **gethostbyname** (string hostname) \linebreak

Restituisce l'indirizzo IP dell'host Internet specificato da *`hostname`*.

Vedere anche gethostbyaddr().

gethostbynamel (PHP 3, PHP 4 >= 4.0.0)

Ottiene la lista degli indirizzi IP corrispondenti a un dato hostname Internet

array **gethostbynamel** (string hostname) \linebreak

Restituisce una lista di indirizzi IP che risolvono nei confronti dell'host Internet specificato da *`hostname`*.

Vedere anche gethostbyname(), gethostbyaddr(), checkdnsrr(), getmxrr() e la pagina man `named(8)`.

getmxrr (PHP 3, PHP 4 >= 4.0.0)

Ottiene i record MX corrispondenti a un dato nome di host Internet

int **getmxrr** (string hostname, array mxhosts [, array weight]) \linebreak

Cerca nel DNS i record MX corrispondenti a *`hostname`*. Restituisce `TRUE` se ne vengono trovati. Restituisce `FALSE` se non ne vengono trovati o se avviene un errore.

La lista di record MX trovati viene messa nell'array *`mxhosts`*. Se viene indicato l'array *`weight`*, esso viene riempito con le informazioni ottenute sui vari pesi.

Vedere anche checkdnsrr(), gethostbyname(), gethostbynamel(), gethostbyaddr()e la pagina man `named(8)`.

getprotobyname (PHP 4 >= 4.0.0)

Ottiene il numero del protocollo associato al nome del protocollo

int **getprotobyname** (string nome) \linebreak

getprotobyname() restituisce il numero del protocollo associato al protocollo *`nome`* come in `/etc/protocols`.

Vedere anche: getprotobynumber().

getprotobynumber (PHP 4 >= 4.0.0)

Ottiene il nome del protocollo associato al numero del protocollo

string **getprotobynumber** (int numero) \linebreak

getprotobynumber() restituisce il nome del protocollo associato al protocollo *`numero`* come in `/etc/protocols`.

Vedere anche: getprotobyname().

getservbyname (PHP 4 >= 4.0.0)

Ottiene il numero di porta associato ad un servizio Internet e ad un protocollo

int **getservbyname** (string servizio, string protocollo) \linebreak

getservbyname() restituisce la porta Internet corrispondente a *`servizio`* per il *`protocollo`* specificato come in `/etc/services`. *`protocollo`* può essere sia `"tcp"` che `"udp"` (scritti in minuscolo).

Vedere anche: getservbyport().

getservbyport (PHP 4 >= 4.0.0)

Ottiene il servizio Internet corrispondente ad una porta e ad un protocollo

string **getservbyport** (int porta, string protocollo) \linebreak

getservbyport() restituisce il servizio Internet associato a *`porta`* relativamente al *`protocollo`* specificato come in `/etc/services`. *`protocollo`* può essere sia `"tcp"` che `"udp"` (scritti in minuscolo).

Vedere anche: getservbyname().

ip2long (PHP 4 >= 4.0.0)

Converts a string containing an (IPv4) Internet Protocol dotted address into a proper address. Converte una stringa contenente un indirizzo di rete del Protocollo Internet (IPv4) in un indirizzo espresso come tipo di dato int.

int **ip2long** (string indirizzo_ip) \linebreak

La funzione **ip2long()** genera un indirizzo di rete Internet IPv4 a partire dalla rappresentazione in formato standard (stringa separata da punti).

Esempio 1. Esempio di ip2long()

```
<?php
$ip = gethostbyname("www.php.net");
$out = "I seguenti URL sono equivalenti:<br>\n";
$out .= "http://www.php.net/, http://".$ip."/, e http://".sprintf("%u",ip2long($ip))."/<b
echo $out;
?>
```

> **Nota:** Poiché il tipo di dato integer in PHP è signed e molti indirizzi IP risulterebbero essere interi negativi, è necessario usare il formattatore "%u" della funzione sprintf() e printf() per ottenere la rappresentazione in stringa dell'indirizzo IP in modo nsigned.

Questo secondo esempio mostra come stampare un indirizzo convertito, usando la funzione printf():

Esempio 2. Visualizzazione di un indirizzo IP

```
<?php
$ip = gethostbyname("www.php.net");
printf("%u\n", ip2long($ip));
echo $out;
?>
```

Vedere anche: long2ip()

long2ip (PHP 4 >= 4.0.0)

Converte un indirizzo di rete del Protocollo Internet (IPv4) in una stringa contenente un indirizzo espresso secondo la notazione standard di Internet.

string **long2ip** (int proper_address) \linebreak

La funzione **long2ip()** genera un indirizzo Internet in formato separato da punti (es.: aaa.bbb.ccc.ddd) a partire dalla rappresentazione propria.

Vedere anche: ip2long()

openlog (PHP 3, PHP 4 >= 4.0.0)

Apre una connessione al logger di sistema

int **openlog** (string ident, int option, int facility) \linebreak

openlog() apre una connessione al logger di sistema per un programma. La stringa *`ident`* viene aggiunta a ogni messaggio. Valori per *`option`* e *`facility`* sono dati di seguito. L'argomento *`option`* viene usato per indicare quali opzioni di loggin verranno usate durante la generazione di un messaggio di log. L'argomento *`facility`* viene usato per specificare quale tipo di programma sta loggando il messaggio. Questo permette di specificare (nella comfigurazione del syslog della macchina) come trattare i messaggi provenienti dalle diverse facility. L'uso di **openlog()** è opzionale. Viene chiamato automaticamente da syslog() se necessario, in tal caso *`ident`* sarà di default `FALSE`.

Costante	Descrizione

Tabella 1. Opzioni di openlog()

Costante	Descrizione
LOG_CONS	se si verifica un errore durante l'invio dei dati al logger di sistema, scrive direttamente sulla console di sistema
LOG_NDELAY	apre immediatamente una connessione al logger
LOG_ODELAY	(default) ritarda l'apertura della connessione fino a quando non viene loggato il primo messaggio
LOG_PERROR	stampa un messaggio di log anche su standard error
LOG_PID	include il PID in ciascun messaggio

Si possono usare una o più di queste opzioni. Usando opzioni multiple è necessario usare `OR`, ad esempio per aprire la connessione immediatamente, scrivere sulla console e il PID in ciascun messaggio, si dovrà usare: `LOG_CONS | LOG_NDELAY | LOG_PID`

Tabella 2. Facility di openlog()

Costante	Descrizione
LOG_AUTH	messaggi di sicurezza/autorizzazione (usa LOG_AUTHPRIV nei sistemi dove è definita quella costante)
LOG_AUTHPRIV	messaggi di sicurezza/autorizzazione (private)
LOG_CRON	clock daemon (cron e at)
LOG_DAEMON	altri demoni di sistema
LOG_KERN	messaggi del kernel
LOG_LOCAL0 ... LOG_LOCAL7	riservato per il locale
LOG_LPR	sottosistema line printer
LOG_MAIL	sottosistema mail
LOG_NEWS	sottosistema news di USENET
LOG_SYSLOG	messaggi generati internamente da syslogd
LOG_USER	messaggi generici user-level
LOG_UUCP	sottosistema UUCP

Vedere anche define_syslog_variables(), syslog() e closelog().

pfsockopen (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Apre una connessione persistente Internet o di tipo domain socket Unix

int **pfsockopen** (string hostname, int port [, int errno [, string errstr [, int timeout]]]) \linebreak

Questa funzione si comporta esattamente come fsockopen() con la differenza che la connessione non viene chiusa dopo che lo script ha finito la sua esecuzione. È la versione persistente di fsockopen().

socket_get_status (PHP 4 >= 4.0.0)

Restituisce informazioni su una risorsa socket esistente

array **socket_get_status** (resource socket_get_status) \linebreak

Restituisce informazioni su una risorsa socket esistente. Attualmente restituisce quattro campi nell'array risultante:

- *timed_out* (bool) - Il socket è andato in time out aspettando i dati
- *blocked* (bool) - Il socket è bloccato
- *eof* (bool) - Indica l'evento EOF
- *unread_bytes* (int) - Numero di byte rimasti nel buffer del socket

Vedere anche: socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_strerror() e la Socket extension.

socket_set_blocking (PHP 4 >= 4.0.0)

Imposta un socket in modalità blocking/non-blocking

int **socket_set_blocking** (int descrittore del socket, int modo) \linebreak

Se *modo* è `FALSE`, il dato descrittore del socket verrà cambiato in modalità non-blocking e se `TRUE`, verrà cambiato in modalità blocking. Questo ha effetti su chiamate tipo quelle di fgets() che leggono dirttamente dal socket. In modo non-blocking una chiamata a fgets() restituirà sempre subito i dati, mentre in modalità blocking, essa attenderà che i dati siano disponibili nel socket.

Questa funzione era precedentemente chiamata **set_socket_blocking()**, ma questo uso è deprecato.

socket_set_timeout (PHP 4 >= 4.0.0)

Imposta il tempo di timeout per un socket

bool **socket_set_timeout** (int descrittore socket, int secondi, int microsecondi) \linebreak

Imposta il valore di timeout per un *descrittore socket*, espresso come la somma di *secondi* e *microsecondi*.

Esempio 1. Esempio di socket_set_timeout()

```
<?php
$fp = fsockopen("www.php.net", 80);
if(!$fp) {
```

```
    echo "Impossibile aprire\n";
} else {
    fputs($fp,"GET / HTTP/1.0\n\n");
    $start = time();
    socket_set_timeout($fp, 2);
    $res = fread($fp, 2000);
    var_dump(socket_get_status($fp));
    fclose($fp);
    print $res;
}
?>
```

Questa funzione era precedentemente chiamata **set_socket_timeout()**, ma questo uso è deprecato.

Vedere anche: fsockopen() e fopen().

syslog (PHP 3, PHP 4 >= 4.0.0)

Genera un messaggio del system log

int **syslog** (int priorità, string messaggio) \linebreak

syslog() genera un messaggio di log che viene distribuito dal logger di sistema. `priorità` è la combinazione della facility e del livello, valori utilizzabili sono riportati nella prossima sezione. L'argomento rimanente è il messaggio da inviare, eccetto i due caratteri `%m` che vengono sostituiti dalla stringa del messaggio di errore (strerror) corrispondente all'attuale valore di errno.

Tabella 1. Priorità syslog() (in ordine discendente)

Costante	Descrizione
LOG_EMERG	sistema non utilizzabile
LOG_ALERT	azione da intraprendere immediatamente
LOG_CRIT	condizioni critiche
LOG_ERR	condizioni di errore
LOG_WARNING	condizioni di attenzione
LOG_NOTICE	condizione normale, ma significativa
LOG_INFO	messaggio di informazione
LOG_DEBUG	messaggio a livello di debug

Esempio 1. Uso di syslog()

```
<?php
define_syslog_variables();
// apre il syslog, include l'ID del processo, invia il
```

```
// log anche su standard error e fa uso di un meccanismo
// di logging defiito dall'utente
openlog("IlMioLog", LOG_PID | LOG_PERROR, LOG_LOCAL0);

// un po' di codice

if (client_autorizzato()) {
    // fa qualcosa
} else {
    // client non autorizzato!
    // logga il tentativo
    $accesso = date("Y/m/d H:i:s");
    syslog(LOG_WARNING,"Client non autorizzato: $accesso $REMOTE_ADDR ($HTTP_USER_AGENT)"
}

closelog();
?>
```

Per informazioni su come creare un gestore di log definito dall'utente, fare riferimento alla man page syslog.conf(5) di Unix. Ulteriori informazioni sulle facility di syslog e sulle sue opzioni possono essere trovate sulle macchine Unix nelle man page di syslog(3).

Su Windows NT, il servizio syslog è emulato usando Event Log.

Vedere anche define_syslog_variables(), openlog() e closelog().

LXVI. Ncurses terminal screen control functions

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

What is ncurses?

ncurses (new curses) is a free software emulation of curses in System V Rel 4.0 (and above). It uses terminfo format, supports pads, colors, multiple highlights, form characters and function key mapping.

Platforms

Ncurses is available for the following platforms:

- AIX
- BeOS
- Cygwin
- Digital Unix (aka OSF1)
- FreeBSD
- GNU/Linux
- HPUX
- IRIX
- OS/2
- SCO OpenServer
- Solaris
- SunOS

Requirements

You need the ncurses libraries and headerfiles. Download the latest version from the

ftp://ftp.gnu.org/pub/gnu/ncurses/ or from an other GNU-Mirror.

Installation

To get these functions to work, you have to compile the CGI version of PHP with `--with-ncurses`.

Ncurses predefined constants

Error codes

On error ncurses functions return NCURSES_ERR.

Colors

Tabella 1. ncurses color constants

constant	meaning
NCURSES_COLOR_BLACK	no color (black)
NCURSES_COLOR_WHITE	white
NCURSES_COLOR_RED	red - supported when terminal is in color mode
NCURSES_COLOR_GREEN	green - supported when terminal is in color mod
NCURSES_COLOR_YELLOW	yellow - supported when terminal is in color mod
NCURSES_COLOR_BLUE	blue - supported when terminal is in color mod
NCURSES_COLOR_CYAN	cyan - supported when terminal is in color mod
NCURSES_COLOR_MAGENTA	magenta - supported when terminal is in color mod

Keys

Tabella 2. ncurses key constants

constant	meaning
NCURSES_KEY_F0 - NCURSES_KEY_F64	function keys F1 - F64
NCURSES_KEY_DOWN	down arrow
NCURSES_KEY_UP	up arrow
NCURSES_KEY_LEFT	left arrow
NCURSES_KEY_RIGHT	right arrow
NCURSES_KEY_HOME	home key (upward+left arrow)
NCURSES_KEY_BACKSPACE	backspace
NCURSES_KEY_DL	delete line

constant	meaning
NCURSES_KEY_IL	insert line
NCURSES_KEY_DC	delete character
NCURSES_KEY_IC	insert char or enter insert mode
NCURSES_KEY_EIC	exit insert char mode
NCURSES_KEY_CLEAR	clear screen
NCURSES_KEY_EOS	clear to end of screen
NCURSES_KEY_EOL	clear to end of line
NCURSES_KEY_SF	scroll one line forward
NCURSES_KEY_SR	scroll one line backward
NCURSES_KEY_NPAGE	next page
NCURSES_KEY_PPAGE	previous page
NCURSES_KEY_STAB	set tab
NCURSES_KEY_CTAB	clear tab
NCURSES_KEY_CATAB	clear all tabs
NCURSES_KEY_SRESET	soft (partial) reset
NCURSES_KEY_RESET	reset or hard reset
NCURSES_KEY_PRINT	print
NCURSES_KEY_LL	lower left
NCURSES_KEY_A1	upper left of keypad
NCURSES_KEY_A3	upper right of keypad
NCURSES_KEY_B2	center of keypad
NCURSES_KEY_C1	lower left of keypad
NCURSES_KEY_C3	lower right of keypad
NCURSES_KEY_BTAB	back tab
NCURSES_KEY_BEG	beginning
NCURSES_KEY_CANCEL	cancel
NCURSES_KEY_CLOSE	close
NCURSES_KEY_COMMAND	cmd (command)
NCURSES_KEY_COPY	copy
NCURSES_KEY_CREATE	create
NCURSES_KEY_END	end
NCURSES_KEY_EXIT	exit
NCURSES_KEY_FIND	find
NCURSES_KEY_HELP	help
NCURSES_KEY_MARK	mark
NCURSES_KEY_MESSAGE	message
NCURSES_KEY_MOVE	move
NCURSES_KEY_NEXT	next
NCURSES_KEY_OPEN	open
NCURSES_KEY_OPTIONS	options
NCURSES_KEY_PREVIOUS	previous
NCURSES_KEY_REDO	redo

constant	meaning
NCURSES_KEY_REFERENCE	ref (reference)
NCURSES_KEY_REFRESH	refresh
NCURSES_KEY_REPLACE	replace
NCURSES_KEY_RESTART	restart
NCURSES_KEY_RESUME	resume
NCURSES_KEY_SAVE	save
NCURSES_KEY_SBEG	shiftet beg (beginning)
NCURSES_KEY_SCANCEL	shifted cancel
NCURSES_KEY_SCOMMAND	shifted command
NCURSES_KEY_SCOPY	shifted copy
NCURSES_KEY_SCREATE	shifted create
NCURSES_KEY_SDC	shifted delete char
NCURSES_KEY_SDL	shifted delete line
NCURSES_KEY_SELECT	select
NCURSES_KEY_SEND	shifted end
NCURSES_KEY_SEOL	shifted end of line
NCURSES_KEY_SEXIT	shifted exit
NCURSES_KEY_SFIND	shifted find
NCURSES_KEY_SHELP	shifted help
NCURSES_KEY_SHOME	shifted home
NCURSES_KEY_SIC	shifted input
NCURSES_KEY_SLEFT	shifted left arrow
NCURSES_KEY_SMESSAGE	shifted message
NCURSES_KEY_SMOVE	shifted move
NCURSES_KEY_SNEXT	shifted next
NCURSES_KEY_SOPTIONS	shifted options
NCURSES_KEY_SPREVIOUS	shifted previous
NCURSES_KEY_SPRINT	shifted print
NCURSES_KEY_SREDO	shifted redo
NCURSES_KEY_SREPLACE	shifted replace
NCURSES_KEY_SRIGHT	shifted right arrow
NCURSES_KEY_SRSUME	shifted resume
NCURSES_KEY_SSAVE	shifted save
NCURSES_KEY_SSUSPEND	shifted suspend
NCURSES_KEY_UNDO	undo
NCURSES_KEY_MOUSE	mouse event has occured
NCURSES_KEY_MAX	maximum key value

Mouse

Tabella 3. mouse constants

Constant	meaning
NCURSES_BUTTON1_RELEASED - NCURSES_BUTTON4_RELEASED	button (1-4) released
NCURSES_BUTTON1_PRESSED - NCURSES_BUTTON4_PRESSED	button (1-4) pressed
NCURSES_BUTTON1_CLICKED - NCURSES_BUTTON4_CLICKED	button (1-4) clicked
NCURSES_BUTTON1_DOUBLE_CLICKED - NCURSES_BUTTON4_DOUBLE_CLICKED	button (1-4) double clicked
NCURSES_BUTTON1_TRIPLE_CLICKED - NCURSES_BUTTON4_TRIPLE_CLICKED	button (1-4) triple clicked
NCURSES_BUTTON_CTRL	ctrl pressed during click
NCURSES_BUTTON_SHIFT	shift pressed during click
NCURSES_BUTTON_ALT	alt pressed during click
NCURSES_ALL_MOUSE_EVENTS	report all mouse events
NCURSES_REPORT_MOUSE_POSITION	report mouse position

ncurses_addch (PHP 4 >= 4.1.0)

Add character at current position and advance cursor

int **ncurses_addch** (int ch) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_addchnstr (PHP 4 >= 4.2.0)

Add attributed string with specified length at current position

int **ncurses_addchnstr** (string s, int n) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_addchstr (PHP 4 >= 4.2.0)

Add attributed string at current position

int **ncurses_addchstr** (string s) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_addnstr (PHP 4 >= 4.2.0)

Add string with specified length at current position

int **ncurses_addnstr** (string s, int n) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_addstr (PHP 4 >= 4.2.0)

Output text at current position

int **ncurses_addstr** (string text) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_assume_default_colors (PHP 4 >= 4.2.0)

Define default colors for color 0

int **ncurses_assume_default_colors** (int fg, int bg) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_attroff (PHP 4 >= 4.1.0)

Turn off the given attributes

int **ncurses_attroff** (int attributes) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_attron (PHP 4 >= 4.1.0)

Turn on the given attributes

int **ncurses_attron** (int attributes) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_attrset (PHP 4 >= 4.1.0)

Set given attributes

int **ncurses_attrset** (int attributes) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_baudrate (PHP 4 >= 4.1.0)

Returns baudrate of terminal

int **ncurses_baudrate** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_beep (PHP 4 >= 4.1.0)

Let the terminal beep

int **ncurses_beep** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_beep() sends an audlible alert (bell) and if its not possible flashes the screen. Returns FALSE on success, otherwise TRUE.

See also: ncurses_flash()

ncurses_bkgd (PHP 4 >= 4.1.0)

Set background property for terminal screen

int **ncurses_bkgd** (int attrchar) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_bkgdset (PHP 4 >= 4.1.0)

Control screen background

void **ncurses_bkgdset** (int attrchar) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_border (PHP 4 >= 4.2.0)

Draw a border around the screen using attributed characters

int **ncurses_border** (int left, int right, int top, int bottom, int tl_corner, int tr_corner, int bl_corner, int br_corner) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_can_change_color (PHP 4 >= 4.1.0)

Check if we can change terminals colors

bool **ncurses_can_change_color** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

The function **ncurses_can_change_color()** returns `TRUE` or `FALSE`, depending on whether the terminal has color capabilities and whether the programmer can change the colors.

ncurses_cbreak (PHP 4 >= 4.1.0)

Switch of input buffering

bool **ncurses_cbreak** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_cbreak() disables line buffering and character processing (interrupt and flow control characters are unaffected), making characters typed by the user immediately available to the program.

ncurses_cbreak() returns TRUE or NCURSES_ERR if any error occured.

See also: ncurses_nocbreak()

ncurses_clear (PHP 4 >= 4.1.0)

Clear screen

bool **ncurses_clear** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_clear() clears the screen completely without setting blanks. Returns FALSE on success, otherwise TRUE.

Note: **ncurses_clear()** clears the screen without setting blanks, which have the current background rendition. To clear screen with blanks, use ncurses_erase().

See also: ncurses_erase()

ncurses_clrtobot (PHP 4 >= 4.1.0)

Clear screen from current position to bottom

bool **ncurses_clrtobot** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_clrtobot() erases all lines from cursor to end of screen and creates blanks. Blanks created by **ncurses_clrtobot()** have the current background rendition. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_clear(), ncurses_clrtoeol()

ncurses_clrtoeol (PHP 4 >= 4.1.0)

Clear screen from current position to end of line

bool **ncurses_clrtoeol** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_clrtoeol() erases the current line from cursor position to the end. Blanks created by **ncurses_clrtoeol()** have the current background rendition. Returns TRUE if any error occured, otherwise FALSE.

See also: ncurses_clear(), ncurses_clrtobot()

ncurses_color_set (PHP 4 >= 4.1.0)

Set fore- and background color

int **ncurses_color_set** (int pair) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_curs_set (PHP 4 >= 4.1.0)

Set cursor state

int **ncurses_curs_set** (int visibility) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_def_prog_mode (PHP 4 >= 4.1.0)

Saves terminals (program) mode

bool **ncurses_def_prog_mode** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_def_prog_mode() saves the current terminal modes for program (in curses) for use by **ncurses_reset_prog_mode()**. Returns `FALSE` on success, otherwise `TRUE`.

See also: **ncurses_reset_prog_mode()**

ncurses_def_shell_mode (PHP 4 >= 4.1.0)

Saves terminals (shell) mode

bool **ncurses_def_shell_mode** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_def_shell_mode() saves the current terminal modes for shell (not in curses) for use by **ncurses_reset_shell_mode()**. Returns `FALSE` on success, otherwise `TRUE`.

See also: **ncurses_reset_shell_mode()**

ncurses_define_key (PHP 4 >= 4.2.0)

Define a keycode

int **ncurses_define_key** (string definition, int keycode) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_delay_output (PHP 4 >= 4.1.0)

Delay output on terminal using padding characters

int **ncurses_delay_output** (int milliseconds) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_delch (PHP 4 >= 4.1.0)

Delete character at current position, move rest of line left

bool **ncurses_delch** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_delch() deletes the character under the cursor. All characters to the right of the cursor on the same line are moved to the left one position and the last character on the line is filled with a blank. The cursor position does not change. Returns FALSE on success, otherwise TRUE.

See also: ncurses_deleteln()

ncurses_deleteln (PHP 4 >= 4.1.0)

Delete line at current position, move rest of screen up

bool **ncurses_deleteln** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_deleteln() deletes the current line under cursorposition. All lines below the current line are moved up one line. The bottom line of window is cleared. Cursor position does not change. Returns FALSE on success, otherwise TRUE.

See also: ncurses_delch()

ncurses_delwin (PHP 4 >= 4.1.0)

Delete a ncurses window

int **ncurses_delwin** (resource window) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_doupdate (PHP 4 >= 4.1.0)

Write all prepared refreshes to terminal

bool **ncurses_doupdate** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_doupdate()() compares the virtual screen to the physical screen and updates the physical screen. This way is more effective than using multiple refresh calls. Returns FALSE on success, TRUE if any error occured.

ncurses_echo (PHP 4 >= 4.1.0)

Activate keyboard input echo

bool **ncurses_echo** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_echo() enables echo mode. All characters typed by user are echoed by ncurses_getch(). Returns FALSE on success, TRUE if any error occured.

To disable echo mode use ncurses_noecho().

ncurses_echochar (PHP 4 >= 4.1.0)

Single character output including refresh

int **ncurses_echochar** (int character) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_end (PHP 4 >= 4.1.0)

Stop using ncurses, clean up the screen

int **ncurses_end** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_erase (PHP 4 >= 4.1.0)

Erase terminal screen

bool **ncurses_erase** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_erase() fills the terminal screen with blanks. Created blanks have the current background rendition, set by ncurses_bkgd(). Returns `FALSE` on success, `TRUE` if any error occured.

See also: ncurses_bkgd(), ncurses_clear()

ncurses_erasechar (PHP 4 >= 4.1.0)

Returns current erase character

string **ncurses_erasechar** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_erasechar() returns the current erase char character.

See also: ncurses_killchar()

ncurses_filter (PHP 4 >= 4.1.0)

int **ncurses_filter** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_flash (PHP 4 >= 4.1.0)

Flash terminal screen (visual bell)

bool **ncurses_flash** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_flash() flashes the screen, and if its not possible, sends an audible alert (bell). Returns `FALSE` on success, otherwise `TRUE`.

See also: ncurses_beep()

ncurses_flushinp (PHP 4 >= 4.1.0)

Flush keyboard input buffer

bool **ncurses_flushinp** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

The **ncurses_flushinp()** throws away any typeahead that has been typed and has not yet been read by your program. Returns `FALSE` on success, otherwise `TRUE`.

ncurses_getch (PHP 4 >= 4.1.0)

Read a character from keyboard

int **ncurses_getch** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_getmouse (PHP 4 >= 4.2.0)

Reads mouse event

bool **ncurses_getmouse** (array mevent) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_getmouse() reads mouse event out of queue. Function **ncurses_getmouse()** will return ;`FALSE` if a mouse event is actually visible in the given window, otherwise it will return `TRUE`. Event options will be delivered in parameter *`mevent`*, which has to be an array, passed by reference (see example below). On success an associative array with following keys will be delivered:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

Esempio 1. ncurses_getmouse() example

```
switch (ncurses_getch){
  case NCURSES_KEY_MOUSE:
    if (!ncurses_getmouse(&$mevent)){
```

```
      if ($mevent["mmask"] & NCURSES_MOUSE_BUTTON1_PRESSED){
        $mouse_x = $mevent["x"]; // Save mouse position
        $mouse_y = $mevent["y"];
      }
    }
  break;

  default:
    ....
}
```

See also: ncurses_ungetmouse()

ncurses_halfdelay (PHP 4 >= 4.1.0)

Put terminal into halfdelay mode

int **ncurses_halfdelay** (int tenth) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_has_colors (PHP 4 >= 4.1.0)

Check if terminal has colors

bool **ncurses_has_colors** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_has_colors() returns `TRUE` or `FALSE` depending on whether the terminal has color capacitites.

See also: ncurses_can_change_color()

ncurses_has_ic (PHP 4 >= 4.1.0)

Check for insert- and delete-capabilities

bool **ncurses_has_ic** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_has_ic() checks terminals insert- and delete capabilitites. It returns `TRUE` when terminal has insert/delete-capabilities, otherwise `FALSE`.

See also: ncurses_has_il()

ncurses_has_il (PHP 4 >= 4.1.0)

Check for line insert- and delete-capabilities

bool **ncurses_has_il** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_has_il() checks terminals insert- and delete-line-capabilities. It returns `TRUE` when terminal has insert/delete-line capabilities, otherwise `FALSE`

See also: ncurses_has_ic()

ncurses_has_key (PHP 4 >= 4.1.0)

Check for presence of a function key on terminal keyboard

int **ncurses_has_key** (int keycode) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_hline (PHP 4 >= 4.2.0)

Draw a horizontal line at current position using an attributed character and max. n characters long

int **ncurses_hline** (int charattr, int n) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_inch (PHP 4 >= 4.1.0)

Get character and attribute at current position

string **ncurses_inch** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_inch() returns the character from the current position.

ncurses_init (PHP 4 >= 4.1.0)

Initialize ncurses

int **ncurses_init** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_init_color (PHP 4 >= 4.2.0)

Set new RGB value for color

int **ncurses_init_color** (int color, int r, int g, int b) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_init_pair (PHP 4 >= 4.1.0)

Allocate a color pair

int **ncurses_init_pair** (int pair, int fg, int bg) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_insch (PHP 4 >= 4.1.0)

Insert character moving rest of line including character at current position

int **ncurses_insch** (int character) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_insdelln (PHP 4 >= 4.1.0)

Insert lines before current line scrolling down (negative numbers delete and scroll up)

int **ncurses_insdelln** (int count) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_insertln (PHP 4 >= 4.1.0)

Insert a line, move rest of screen down

bool **ncurses_insertln** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_insertln() inserts a new line above the current line. The bottom line will be lost.

ncurses_insstr (PHP 4 >= 4.2.0)

Insert string at current position, moving rest of line right

int **ncurses_insstr** (string text) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_instr (PHP 4 >= 4.2.0)

Reads string from terminal screen

int **ncurses_instr** (string buffer) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_instr() returns the number of charaters read from the current character position until end of line. `buffer` contains the characters. Atrributes are stripped from the characters.

ncurses_isendwin (PHP 4 >= 4.1.0)

Ncurses is in endwin mode, normal screen output may be performed

bool **ncurses_isendwin** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_isendwin() returns `TRUE`, if **ncurses_endwin()** has been called without any subsequent calls to ncurses_wrefresh(), otherwise `FALSE`.

See also: **ncurses_endwin() ncurses_wrefresh()()**

ncurses_keyok (PHP 4 >= 4.2.0)

Enable or disable a keycode

int **ncurses_keyok** (int keycode, bool enable) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_killchar (PHP 4 >= 4.1.0)

Returns current line kill character

bool **ncurses_killchar** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_killchar() returns the current line kill character.

See also: ncurses_erasechar()

ncurses_longname (PHP 4 >= 4.2.0)

Returns terminals description

string **ncurses_longname** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_longname() returns a verbose description of the terminal. The descritpion is truncated to 128 characters. On Error **ncurses_longname()** returns NULL.

See also: ncurses_termname()

ncurses_mouseinterval (PHP 4 >= 4.1.0)

Set timeout for mouse button clicks

int **ncurses_mouseinterval** (int milliseconds) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mousemask (PHP 4 >= 4.2.0)

Sets mouse options

int **ncurses_mousemask** (int newmask, int oldmask) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Function **ncurses_mousemask()** will set mouse events to be reported. By default no mouse events will be reported. The function **ncurses_mousemask()** will return a mask to indicated which of the in parameter *newmask* specified mouse events can be reported. On complete failure, it returns 0. In parameter *oldmask*, which is passed by reference **ncurses_mousemask()** returns the previous value of mouse event mask. Mouse events are represented bei NCURSES_KEY_MOUSE in the **ncurses_wgetch()** input stream. To read the event data and pop the event of of queue, call ncurses_getmouse().

As a side effect, setting a zero mousemask in *newmask* turns off the mouse pointer. Setting a non zero value turns mouse pointer on.

mouse mask options can be set with the following predefined constants:

- NCURSES_BUTTON1_PRESSED
- NCURSES_BUTTON1_RELEASED
- NCURSES_BUTTON1_CLICKED
- NCURSES_BUTTON1_DOUBLE_CLICKED
- NCURSES_BUTTON1_TRIPLE_CLICKED
- NCURSES_BUTTON2_PRESSED
- NCURSES_BUTTON2_RELEASED
- NCURSES_BUTTON2_CLICKED
- NCURSES_BUTTON2_DOUBLE_CLICKED
- NCURSES_BUTTON2_TRIPLE_CLICKED
- NCURSES_BUTTON3_PRESSED
- NCURSES_BUTTON3_RELEASED
- NCURSES_BUTTON3_CLICKED
- NCURSES_BUTTON3_DOUBLE_CLICKED
- NCURSES_BUTTON3_TRIPLE_CLICKED
- NCURSES_BUTTON4_PRESSED
- NCURSES_BUTTON4_RELEASED
- NCURSES_BUTTON4_CLICKED
- NCURSES_BUTTON4_DOUBLE_CLICKED

- NCURSES_BUTTON4_TRIPLE_CLICKED
- NCURSES_BUTTON_SHIFT>
- NCURSES_BUTTON_CTRL
- NCURSES_BUTTON_ALT
- NCURSES_ALL_MOUSE_EVENTS
- NCURSES_REPORT_MOUSE_POSITION

See also: ncurses_getmouse(), ncurses_ungetmouse() **ncurese_getch()**

Esempio 1. ncurses_mousemask() example

```
$newmask = NCURSES_BUTTON1_CLICKED + NCURSES_BUTTON1_RELEASED;
$mask = ncurses_mousemask($newmask, &$oldmask);
if ($mask & $newmask){
  printf ("All specified mouse options will be supported\n");
}
```

ncurses_move (PHP 4 >= 4.1.0)

Move output position

int **ncurses_move** (int y, int x) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvaddch (PHP 4 >= 4.2.0)

Move current position and add character

int **ncurses_mvaddch** (int y, int x, int c) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvaddchnstr (PHP 4 >= 4.2.0)

Move position and add attrributed string with specified length

int **ncurses_mvaddchnstr** (int y, int x, string s, int n) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvaddchstr (PHP 4 >= 4.2.0)

Move position and add attributed string

int **ncurses_mvaddchstr** (int y, int x, string s) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvaddnstr (PHP 4 >= 4.2.0)

Move position and add string with specified length

int **ncurses_mvaddnstr** (int y, int x, string s, int n) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvaddstr (PHP 4 >= 4.2.0)

Move position and add string

int **ncurses_mvaddstr** (int y, int x, string s) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvcur (PHP 4 >= 4.2.0)

Move cursor immediately

int **ncurses_mvcur** (int old_y, int old_x, int new_y, int new_x) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvdelch (PHP 4 >= 4.2.0)

Move position and delete character, shift rest of line left

int **ncurses_mvdelch** (int y, int x) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvgetch (PHP 4 >= 4.2.0)

Move position and get character at new position

int **ncurses_mvgetch** (int y, int x) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvhline (PHP 4 >= 4.2.0)

Set new position and draw a horizontal line using an attributed character and max. n characters long

int **ncurses_mvhline** (int y, int x, int attrchar, int n) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvinch (PHP 4 >= 4.2.0)

Move position and get attributed character at new position

int **ncurses_mvinch** (int y, int x) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvvline (unknown)

Set new position and draw a vertical line using an attributed character and max. n characters long

int **ncurses_mvvline** (int y, int x, int attrchar, int n) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_mvwaddstr (PHP 4 >= 4.2.0)

Add string at new position in window

int **ncurses_mvwaddstr** (resource window, int y, int x, string text) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_napms (PHP 4 >= 4.1.0)

Sleep

int **ncurses_napms** (int milliseconds) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_newwin (PHP 4 >= 4.1.0)

Create a new window

int **ncurses_newwin** (int rows, int cols, int y, int x) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_nl (PHP 4 >= 4.1.0)

Translate newline and carriage return / line feed

bool **ncurses_nl** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_nocbreak (PHP 4 >= 4.1.0)

Switch terminal to cooked mode

bool **ncurses_nocbreak** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_nocbreak() routine returns terminal to normal (cooked) mode. Initially the terminal may or may not in cbreak mode as the mode is inherited. Therefore a program should call ncurses_cbreak() and **ncurses_nocbreak()** explicitly. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_cbreak()

ncurses_noecho (PHP 4 >= 4.1.0)

Switch off keyboard input echo

bool **ncurses_noecho** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_noecho() prevents echoing of user typed characters. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_echo(), ncurses_getch()

ncurses_nonl (PHP 4 >= 4.1.0)

Do not translate newline and carriage return / line feed

bool **ncurses_nonl** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_noqiflush (PHP 4 >= 4.1.0)

Do not flush on signal characters

int **ncurses_noqiflush** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_noraw (PHP 4 >= 4.1.0)

Switch terminal out of raw mode

bool **ncurses_noraw** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_noraw() switches the terminal out of raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_raw(), ncurses_cbreak(), ncurses_nocbreak()

ncurses_putp (PHP 4 >= 4.2.0)

int **ncurses_putp** (string text) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_qiflush (PHP 4 >= 4.1.0)

Flush on signal characters

int **ncurses_qiflush** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_raw (PHP 4 >= 4.1.0)

Switch terminal into raw mode

bool **ncurses_raw** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_raw() places the terminal in raw mode. Raw mode is similar to cbreak mode, in that characters typed are immediately passed through to the user program. The differences that are that in raw mode, the interrupt, quit, suspend and flow control characters are all passed through uninterpreted, instead of generating a signal. Returns `TRUE` if any error occured, otherwise `FALSE`.

See also: ncurses_noraw(), ncurses_cbreak(), ncurses_nocbreak()

ncurses_refresh (PHP 4 >= 4.1.0)

Refresh screen

int **ncurses_refresh** (int ch) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_resetty (PHP 4 >= 4.1.0)

Restores saved terminal state

bool **ncurses_resetty** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Function **ncurses_resetty()** restores the terminal state, which was previously saved by calling ncurses_savetty(). This function always returns FALSE.

See also: ncurses_savetty()

ncurses_savetty (PHP 4 >= 4.1.0)

Saves terminal state

bool **ncurses_savetty** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Function **ncurses_savetty()** saves the current terminal state. The saved terminal state can be restored with function ncurses_resetty(). **ncurses_savetty()** always returns FALSE.

See also: ncurses_resetty()

ncurses_scr_dump (PHP 4 >= 4.2.0)

Dump screen content to file

int **ncurses_scr_dump** (string filename) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_scr_init (PHP 4 >= 4.2.0)

Initialize screen from file dump

int **ncurses_scr_init** (string filename) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_scr_restore (PHP 4 >= 4.2.0)

Restore screen from file dump

int **ncurses_scr_restore** (string filename) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_scr_set (PHP 4 >= 4.2.0)

Inherit screen from file dump

int **ncurses_scr_set** (string filename) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_scrl (PHP 4 >= 4.1.0)

Scroll window content up or down without changing current position

int **ncurses_scrl** (int count) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_slk_attr (PHP 4 >= 4.1.0)

Returns current soft label key attribute

bool **ncurses_slk_attr** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_slk_attr() returns the current soft label key attribute. On error returns `TRUE`, otherwise `FALSE`.

ncurses_slk_attroff (PHP 4 >= 4.1.0)

int **ncurses_slk_attroff** (int intarg) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_slk_attron (PHP 4 >= 4.1.0)

int **ncurses_slk_attron** (int intarg) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_slk_attrset (PHP 4 >= 4.1.0)

int **ncurses_slk_attrset** (int intarg) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_slk_clear (PHP 4 >= 4.1.0)

Clears soft labels from screen

bool **ncurses_slk_clear** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

The function **ncurses_slk_clear()** clears soft label keys from screen. Returns `TRUE` on error, otherwise `FALSE`.

ncurses_slk_color (PHP 4 >= 4.1.0)

Sets color for soft label keys

int **ncurses_slk_color** (int intarg) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_slk_init (PHP 4 >= 4.1.0)

Initializes soft label key functions

bool **ncurses_slk_init** (int format) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Funtion **ncurses_slk_init()** must be called before **ncurses_initscr()** or **ncurses_newterm()** is called. If **ncurses_initscr()** eventually uses a line from stdscr to emulate the soft labels, then *`format`* determines how the labels are arranged of the screen. Setting *`format`* to 0 indicates a 3-2-3 arrangement of the labels, 1 indicates a 4-4 arrangement and 2 indicates the PC like 4-4-4 mode, but in addition an index line will be created.

ncurses_slk_noutrefresh (PHP 4 >= 4.1.0)

Copies soft label keys to virtual screen

bool **ncurses_slk_noutrefresh** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_slk_refresh (PHP 4 >= 4.1.0)

Copies soft label keys to screen

bool **ncurses_slk_refresh** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_slk_refresh() copies soft label keys from virtual screen to physical screen. Returns `TRUE` on error, otherwise `FALSE`.

ncurses_slk_restore (PHP 4 >= 4.1.0)

Restores soft label keys

bool **ncurses_slk_restore** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

The function **ncurses_slk_restore()** restores the soft label keys after ncurses_slk_clear() has been performed.

ncurses_slk_touch (PHP 4 >= 4.1.0)

Fources output when ncurses_slk_noutrefresh is performed

bool **ncurses_slk_touch** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

The **ncurses_slk_touch()** function forces all the soft labels to be output the next time a ncurses_slk_noutrefresh() is performed.

ncurses_standend (PHP 4 >= 4.1.0)

Stop using 'standout' attribute

int **ncurses_standend** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_standout (PHP 4 >= 4.1.0)

Start using 'standout' attribute

int **ncurses_standout** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_start_color (PHP 4 >= 4.1.0)

Start using colors

int **ncurses_start_color** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_termattrs (PHP 4 >= 4.1.0)

Returns a logical OR of all attribute flags supported by terminal

bool **ncurses_termattrs** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_termname (PHP 4 >= 4.2.0)

Returns terminals (short)-name

string **ncurses_termname** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_termname() returns terminals shortname. The shortname is truncated to 14 characters. On error **ncurses_termname()** returns NULL.

See also: ncurses_longname()

ncurses_timeout (PHP 4 >= 4.1.0)

Set timeout for special key sequences

void **ncurses_timeout** (int millisec) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_typeahead (PHP 4 >= 4.1.0)

Specify different filedescriptor for typeahead checking

int **ncurses_typeahead** (int fd) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_ungetch (PHP 4 >= 4.1.0)

Put a character back into the input stream

int **ncurses_ungetch** (int keycode) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_ungetmouse (PHP 4 >= 4.2.0)

Pushes mouse event to queue

bool **ncurses_ungetmouse** (array mevent) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

ncurses_getmouse() pushes a KEY_MOUSE event onto the unput queue and associates with this event the given state sata and screen-relative character cell coordinates, specified in *`mevent`*. Event options will be specified in associative array *`mevent`*:

- "id" : Id to distinguish multiple devices
- "x" : screen relative x-position in character cells
- "y" : screen relative y-position in character cells
- "z" : currently not supported
- "mmask" : Mouse action

ncurses_ungetmouse() returns FALSE on success, otherwise TRUE.

See also: ncurses_getmouse()

ncurses_use_default_colors (PHP 4 >= 4.1.0)

Assign terminal default colors to color id -1

bool **ncurses_use_default_colors** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_use_env (PHP 4 >= 4.1.0)

Control use of environment information about terminal size

void **ncurses_use_env** (bool flag) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_use_extended_names (PHP 4 >= 4.1.0)

Control use of extended names in terminfo descriptions

int **ncurses_use_extended_names** (bool flag) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_vidattr (PHP 4 >= 4.1.0)

int **ncurses_vidattr** (int intarg) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_vline (PHP 4 >= 4.2.0)

Draw a vertical line at current position using an attributed character and max. n characters long

int **ncurses_vline** (int charattr, int n) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

ncurses_wrefresh (PHP 4 >= 4.2.0)

Refresh window on terminal screen

int **ncurses_wrefresh** (resource window) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

undocumented

LXVII. Lotus Notes functions

Attenzione

Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

notes_body (PHP 4 >= 4.0.5)

Open the message msg_number in the specified mailbox on the specified server (leave serv

array **notes_body** (string server, string mailbox, int msg_number) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_copy_db (PHP 4 >= 4.0.5)

Create a note using form form_name

string **notes_copy_db** (string from_database_name, string to_database_name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_create_db (PHP 4 >= 4.0.5)

Create a Lotus Notes database

bool **notes_create_db** (string database_name) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_create_note (PHP 4 >= 4.0.5)

Create a note using form form_name

string **notes_create_note** (string database_name, string form_name) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_drop_db (PHP 4 >= 4.0.5)

Drop a Lotus Notes database

bool **notes_drop_db** (string database_name) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_find_note (PHP 4 >= 4.0.5)

Returns a note id found in database_name. Specify the name of the note. Leaving type bla

bool **notes_find_note** (string database_name, string name [, string type]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_header_info (PHP 4 >= 4.0.5)

Open the message msg_number in the specified mailbox on the specified server (leave serv

object **notes_header_info** (string server, string mailbox, int msg_number) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_list_msgs (PHP 4 >= 4.0.5)

Returns the notes from a selected database_name

bool **notes_list_msgs** (string db) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_mark_read (PHP 4 >= 4.0.5)

Mark a note_id as read for the User user_name

string **notes_mark_read** (string database_name, string user_name, string note_id) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_mark_unread (PHP 4 >= 4.0.5)

Mark a note_id as unread for the User user_name

string **notes_mark_unread** (string database_name, string user_name, string note_id) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_nav_create (PHP 4 >= 4.0.5)

Create a navigator name, in database_name

bool **notes_nav_create** (string database_name, string name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_search (PHP 4 >= 4.0.5)

Find notes that match keywords in database_name

string **notes_search** (string database_name, string keywords) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_unread (PHP 4 >= 4.0.5)

Returns the unread note id's for the current User user_name

string **notes_unread** (string database_name, string user_name) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

notes_version (PHP 4 >= 4.0.5)

Get the version Lotus Notes

string **notes_version** (string database_name) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

LXVIII. Funzioni ODBC Unificate

In aggiunta al normale supporto ODBC, le funzioni ODBC unificate del PHP consentono l'accesso a diversi database che hanno preso in prestito la semantica dell'API ODBC per implementare la loro API. Invece di mantenere più driver per database che sono tutti pressoché identici, questi driver sono stati riuniti in un singolo insieme di funzioni ODBC.

> **Nota:** Nella connessione ai database sopra elencati non vengono coinvolte funzioni ODBC. Le funzioni che vengono utilizzate per collegarsi nativamente con essi condividono solamente lo stesso nome e sintassi delle funzioni ODBC. L'eccezione a questo è iODBC. Compilando il PHP con il supporto di iODBC, si può utilizzare qualsiasi driver compatibile ODBC nelle applicazioni PHP. iODBC è gestito da OpenLink Software (http://www.openlinksw.com/). Maggiori informazioni su iODBC, e un HOWTO sono diponibili nel sito www.iodbc.org (http://www.iodbc.org/).

Requisiti

Le funzioni ODBC Unificate supportano i seguenti database: Adabas D (http://www.software-ag.com/adabasd/), IBM DB2 (http://www.ibm.com/db2/), iODBC (http://www.iodbc.org/), Solid (http://www.solidtech.com/) e Sybase SQL Anywhere (http://www.sybase.com/). Per potere accedere a questi database, occorre avere installato le librerie necessarie.

Installazione

Vedere il capitolo Installazione su Unix Systems per avere maggiori informazioni su come configurare PHP per accedere a questi database.

Configurazione del Runtime

Il comportamento delle funzioni ODBC viene modificato da parametri presenti nel file di configurazione `php.ini`.

Tabella 1. Opzioni di configurazione per ODBC

Nome	Default	Modificabile
odbc.default_db *	NULL	PHP_INI_ALL
odbc.default_user *	NULL	PHP_INI_ALL
odbc.default_pw *	NULL	PHP_INI_ALL
odbc.allow_persistent	"1"	PHP_INI_SYSTEM
odbc.check_persistent	"1"	PHP_INI_SYSTEM
odbc.max_persistent	"-1"	PHP_INI_SYSTEM
odbc.max_links	"-1"	PHP_INI_SYSTEM
odbc.defaultlrl	"4096"	PHP_INI_ALL

Nome	Default	Modificabile
odbc.defaultbinmode	"1"	PHP_INI_ALL

Nota: Le linee marcate con * non sono state ancora implementate.

Per avere maggiori dettagli e le definizioni delle costanti PHP_INI_* vedere ini_set().

Di seguito verranno brevemente illustrate le direttive di configurazione.

`odbc.default_db` string

Sorgente dei dati ODBC da utilizzare se non viene specificato nulla in odbc_connect() o in

odbc_pconnect().

odbc.default_user string

Nome utente da utilizzare se non viene specificato in odbc_connect() o in odbc_pconnect().

odbc.default_pw string

Password da utilizzare se non viene specificata in odbc_connect() o in odbc_pconnect().

odbc.allow_persistent boolean

Abilita o meno le connessioni ODBC persistenti.

odbc.check_persistent boolean

Controlla se una connessione è ancora valida prima del riutilizzo.

odbc.max_persistent integer

Indica il numero massimo di connessioni persistenti ammesse per processo.

odbc.max_links integer

Indica il numero massimo di connessioni ammesse per processo, compresse le persistenti.

odbc.defaultlrl integer

Gestione dei campi di tipo LONG. Specifica il numero di byte da restituire alla variabile.

odbc.defaultbinmode integer

Gestione dei dati binari.

Tipi di risorse

Questa estensione non definisce alcun tipo di risorsa.

Costanti predefinite

Questa estensione non definisce alcuna costante.

odbc_autocommit (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Valorizza il parametro autocommit

bool **odbc_autocommit** (resource id_connessione [, bool OnOff]) \linebreak

Se non viene fornito il parametro *OnOff*, la funzione restituisce lo stato dell'auto-commit per *id_connessione*. Il valore reso è vero se l'autocommit è attivo, altrimenti falso se non è attivato oppure si verifica un errore.

Se il campo *OnOff* è posto a vero, l' auto-commit è abilitato, se è valorizzato a falso l'auto-commit è disabilitato. La funzione restituisce `TRUE` se l'operazione riesce, `FALSE` se si verifica un errore.

Per default, l'auto-commit è abilitato. La disabilitazione dell'auto-commit equivale ad iniziare una transazione.

Vedere inoltre odbc_commit() e odbc_rollback().

odbc_binmode (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gestione delle colonne di dati binari

int **odbc_binmode** (resource id_risultato, int modalità) \linebreak

(Tipi di campi ODBC SQL coinvolti: BINARY, VARBINARY, LONGVARBINARY)

- ODBC_BINMODE_PASSTHRU: Restituzione del dato binario direttamente al client
- ODBC_BINMODE_RETURN: restituisce il dato inalterato
- ODBC_BINMODE_CONVERT: Conversione in char

Quando si esegue la conversione da dati binari SQL a dati di tipo char del C, ciascun byte (8 bits) dei dati sorgenti vengono rappresentati da 2 caratteri ASCII. Questi caratteri sono la rappresentazione ASCII dei numeri nella loro forma esadecimale. Ad esempio, il valore binario 00000001 è convertito in `"01"` e il valore binario 11111111 è convertito come `"FF"`.

Tabella 1. Gestione del tipo LONGVARBINARY

Modalità	Impostazione di longreadlen	Comportamento
ODBC_BINMODE_PASSTHRU	0	direttamente al client
ODBC_BINMODE_RETURN	0	direttamente al client
ODBC_BINMODE_CONVERT	0	direttamente al client
ODBC_BINMODE_PASSTHRU	0	passthru
ODBC_BINMODE_PASSTHRU	>0	direttamente al client
ODBC_BINMODE_RETURN	>0	restituito inalterato
ODBC_BINMODE_CONVERT	>0	restituito come char

Se viene utilizzata odbc_fetch_into(), nei casi in cui il dato viene inviato direttamente al client, quest'ultima restituisce una stringa vuota per le colonne binarie.

Se l'argomento *`id_risultato`* è valorizzato a `0`, il settaggio viene applicato come default per i nuovi risultati.

> **Nota:** I valori di default per longreadlen è `4096`, mentre la modalità di default è `ODBC_BINMODE_RETURN`. La gestione delle colonne di campi long binary, è anche gestita dalla funzione odbc_longreadlen()

odbc_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Chiude una connessione ODBC

void **odbc_close** (resource id_connessione) \linebreak

odbc_close() chiude la connessione con il database server associata all'identificativo di connessione indicato.

> **Nota:** Se ci sono delle transazioni aperte sulla connessione richiesta, la funzione fallisce. In questo caso la connessione resta aperta.

odbc_close_all (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Chiude tutte le connessioni ODBC

void **odbc_close_all** (void) \linebreak

La funzione **odbc_close_all()** chiude tutte le connessioni aperte con il database server

> **Nota:** Se ci sono delle transazioni aperte sulla connessione richiesta, la funzione fallisce. In questo caso la connessione resta aperta.

odbc_columnprivileges (PHP 4 >= 4.0.0)

Restituisce un identificatore di risultato che permette di ricavare l'elenco delle colonne e dei privilegi ad esse associati.

int **odbc_columnprivileges** (resource id_connessione [, string qualifica [, string proprietario [, string nome_tabella [, string nome_colonna]]]]) \linebreak

Elenca le colonne e i privilegi associati ad esse per la tabella data. La funzione ritorna un identificatore di risultato ODBC oppure `FALSE` se si verifica un errore.

Le righe risultanti dall'elaborazione contengono i seguenti campi:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

I campi di ordinamento delle righe risultanti sono TABLE_QUALIFIER, TABLE_OWNER e TABLE_NAME.

L'argomento *`nome_colonna`* accetta dei criteri di ricerca ('%' per indicare zero o più caratteri e '_' per indicare un singolo carattere).

odbc_columns (PHP 4 >= 4.0.0)

Elenca i nomi delle colonne nella tabella specificata. La funzione ritorna un identificatore di risultato contenenti le informazioni.

int **odbc_columns** (resource id_connessione [, string qualifica [, string proprietario [, string nome_tabella [, string nome_colonna]]]]) \linebreak

Elenca i nomi di tutte le colonne presenti nel range richiesto. La funzione restituisce un identificatore di risultato ODBC oppure `FALSE` se si verifica un errore.

Le righe risultanti dall'elaborazione contengono i seguenti campi:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX

- NULLABLE
- REMARKS

I campi di ordinamento delle righe risultanti sono TABLE_QUALIFIER, TABLE_OWNER e TABLE_NAME.

Gli argomenti *proprietario*, *nome_tabella* e *nome_colonna* accettano dei criteri di ricerca ('%' per indicare zero o più caratteri e '_' per indicare un singolo carattere).

Vedere anche odbc_columnprivileges() per ottenere i privilegi associati alle colonne.

odbc_commit (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Esegue una transazione ODBC

bool **odbc_commit** (resource id_connessione) \linebreak

Restituisce: `TRUE` per operazione corretta, `FALSE` se si verifica un errore. Sono eseguite tutte le transazioni pendenti sulla connessione indicata dall'argomento *id_connessione*.

odbc_connect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Apre una connessione con una fonte di dati

resource **odbc_connect** (string dsn, string utente, string password [, int tipo_cursore]) \linebreak

Restituisce un identificatore di connessione ODBC oppure 0 (`falso`) se si verifica un errore.

L'identificatore di connessione ritornato da questa funzione è utilizzato dalle altre funzioni ODBC. Si possono avere più connessioni aperte contemporaneamente. Il quarto parametro (opzionale), setta il tipo di cursore da utilizzare per questa connessione. Normalmente questo parametro non è necessario, ma può essere utilizzato per aggirare dei problemi che si manifestano con alcuni driver ODBC.

Con alcuni driver ODBC, l'esecuzione di complesse procedure può generare un errore tipo: "Non si riesce ad aprire un cursore sulla procedura che richieda qualsiasi cosa oltre ad un singola istruzione select". L'uso di SQL_CUR_USE_ODBC, può evitare questo errore. Inoltre alcuni driver non supportano il parametro row_number della funzione odbc_fetch_row(). In questo caso SQL_CUR_USE_ODBC può essere d'aiuto.

Il campo tipo_cursore può assumere le seguenti costanti:

- SQL_CUR_USE_IF_NEEDED
- SQL_CUR_USE_ODBC
- SQL_CUR_USE_DRIVER
- SQL_CUR_DEFAULT

Per le connessioni persistenti vedere odbc_pconnect().

odbc_cursor (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce il nome del cursore

string **odbc_cursor** (resource id_risultato) \linebreak

odbc_cursor restituisce il nome del cursore per l'argomento id_risultato.

odbc_do (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sinonimo di odbc_exec()

resource **odbc_do** (resource id_connessione, string query) \linebreak

odbc_do() esegue una query sulla connessione data.

odbc_error (PHP 4 >= 4.0.5)

Restituisce l'ultimo codice di errore

string **odbc_error** ([resource id_connessione]) \linebreak

La funzione restituisce un codice di 6 cifre indicante lo stato di ODBC. Se non vi sono errori viene restituita una stringa vuota. Se viene passato il parametro *`id_connessione`*, viene restituito l'ultimo stato di questa connessione, altrimenti si ha l'ultimo stato dell'ultima operazione su una qualsiasi connessione.

Vedere anche: odbc_errormsg() e odbc_exec().

odbc_errormsg (PHP 4 >= 4.0.5)

Restituisce l'ultimo messaggio d'errore

string **odbc_errormsg** ([resource id_connessione]) \linebreak

la funzione restituisce una stringa contenente l'ultimo messaggio di errore generato da ODBC, oppure una stringa vuota se non ci sono errori. Se viene passato il parametro *`id_connessione`*, viene restituito l'ultimo stato di questa connessione, altrimenti si ha l'ultimo stato dell'ultima operazione su una qualsiasi connessione.

Vedere anche: odbc_error() and odbc_exec().

odbc_exec (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Prepara ed esegue una espressione SQL

resource **odbc_exec** (resource id_connessione, string testo_query) \linebreak

Restituisce `FALSE` se si verifica un errore. Restituisce un identificatore del risultato ODBC se l'espressione SQL viene eseguita correttamente.

odbc_exec() invia una espressione SQL al server tramite la connessione specificata da *`id_connessione`*. Questo parametro deve essere un identificativo valido restituito da odbc_connect() oppure odbc_pconnect().

Vedere anche: odbc_prepare() e odbc_execute() per l'esecuzione di molteplici espressioni SQL.

odbc_execute (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Esecuzione di un'espressione memorizzata

resource **odbc_execute** (resource id_risultato [, array array_parametri]) \linebreak

Esegue una espressione SQL memorizzata tramite la funzione odbc_prepare(). Restituisce `TRUE` se l'esecuzione riesce, `FALSE` in caso negativo. L'array *`array_parametri`* occorre soltanto se è necessario fornire parametri all'espressione.

I parametri in *`array_parametri`* saranno sostituiti dai segnaposto nell'ordine dell'espressione predisposta.

Qualsiasi parametro in *`array_parametri`* che inizia e finisce con gli apici singoli sarà considerato come il nome di un file da leggere e inviare al database server come dati per gli appropriati segnaposto.

Nota: Dalla versione 4.1.1 la funzionalità di lettura del file ha le seguenti restrizioni:

- La lettura del file *not* è soggetta alle restrizioni di safe mode o safe mode. Questo sarà risolto nella versione 4.2.0 di PHP.
- File remoti non sono supportati.
- Se si desidera archiviare una stringa che inizia e termina con l'apice singolo, occorre farla precedere dal carattere di escape, oppure da un'altro carattere all'inizio o alla fine del parametro, per prevenire che il parametro sia considerato come nome di un file. Se questo non è possibile occorre utilizzare altri meccanismi per archiviare la stringa, quali, ad esempio, eseguire direttamente la query con odbc_exec().

odbc_fetch_array (PHP 4 >= 4.0.2)

Restituisce una riga in un array associativo

array **odbc_fetch_array** (resource resultato [, int numero_riga]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

odbc_fetch_into (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Scarica una riga del risultato della query in un array

bool **odbc_fetch_into** (resource id_risultato [, int numero_riga, array array_dati]) \linebreak resource **odbc_fetch_into** (resource id_risultato, array array_dati [, int numero_riga]) \linebreak

La funzione restituisce il numero di colonne presenti nel risultato; `FALSE` se si verifica un errore. Il parametro *`array_dati`* deve essere passato per referenza, ma può essere di qualsiasi tipo dato che verrà convertito in array. Nell'array saranno posti i valori delle colonne di una riga tratta dalla tabella risultante dalla query a partire dall'indice 0.

A partire dalla versione 4.0.5 di PHP non occorre passare il parametro *`array_dati`* per riferimento.

A partire dall versione 4.0.6 di PHP il parametro *`numero_riga`* non può essere passato come costante, ma come variabile.

A partire dalla versione 4.2.0 di PHP i parametri *`array_dati`* e *`numero_riga`* sono stati invertiti. Questo permette al parametro numero_riga di essere ancora una costante. Questa è l'ultima variazione alla funzione.

Esempio 1. odbc_fetch_into() Esempi pre 4.0.6

```
$rc = odbc_fetch_into($id_risultato, $array_di_test);
```

oppure

```
$rc = odbc_fetch_into($id_risultato, $riga, $array_di_test);

$rc = odbc_fetch_into($id_risultato, 1, $array_di_test);
```

Esempio 2. odbc_fetch_into() Esempi con PHP 4.0.6

```
$rc = odbc_fetch_into($id_risultato, $array_di_test);
```

oppure

```
$riga = 1;
$rc = odbc_fetch_into($id_risultato, $riga, $array_di_test);
```

Esempio 3. Esempio di odbc_fetch_into()con PHP 4.2.0

```
$rc = odbc_fetch_into($res_id, $my_array);
```

oppure

```
$rc = odbc_fetch_into($res_id,$my_array, 2);
```

odbc_fetch_object (PHP 4 >= 4.0.2)

Restituisce una riga di risultato come un oggetto

object **odbc_fetch_object** (resource risultato [, int NumeroRiga]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

odbc_fetch_row (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Estrae una riga

bool **odbc_fetch_row** (resource id_risultato [, int numero_riga]) \linebreak

Se **odbc_fetch_row()** ha successo (c'è almeno una riga), la funzione restituisce `TRUE`. Altrimenti, se non vi sono più righe, la funzione restituisce `FALSE`.

odbc_fetch_row() estrae un record dai dati restituiti dalle funzioni odbc_do() / odbc_exec(). Dopo l'esecuzione di **odbc_fetch_row()**, i campi della riga sono accessibili tramite la funzione odbc_result().

Se non viene specificato il parametro *numero_riga*, **odbc_fetch_row()** restituisce la riga successiva dal set delle righe risultanti dalla query. Si può intercalare esecuzioni successive di **odbc_fetch_row()** con e senza il parametro *numero_riga*.

Per spostarsi attraverso le righe risultanti, si può eseguire **odbc_fetch_row()** con il parametro *numero_riga* impostato a 1, e quindi continuare ad utilizzare **odbc_fetch_row()** senza *numero_riga*. Se il driver non supporta l'estrazione di una riga per numero, il campo *numero_riga* sarà ignorato.

odbc_field_len (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce la dimensione (precisione) di un campo

int **odbc_field_len** (resource id_risultato, int numero_campo) \linebreak

All'interno di un set di righe, referenziate dall'identificativo di risultato ODBC fornito, la funzione **odbc_field_len()** restituisce la dimensione (precisione) del campo indicato dall'argomento. La numerazione dei campi parte da 1.

Vedere anche: odbc_field_scale() per ottenere la scala di un numero in virgola mobile.

odbc_field_name (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce il nome della colonna

string **odbc_field_name** (resource id_risultato, int numero_campo) \linebreak

La funzione **odbc_field_name()** restituisce il nome del campo presente nella colonna richiesta all'interno di un risultato ODBC identificato dal'argomento id_risultato. La numerazione delle colonne parte da 1. La funzione restituisce `falso` se si verifica un errore.

odbc_field_num (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce il numero di colonna

int **odbc_field_num** (resource id_risultato, string nome_campo) \linebreak

odbc_field_num() restituisce il numero della colonna in cui si trova il campo richiesto all'interno di un risultato ODBC indicato dall'argomento id_risultato. La numerazione delle colonne parte da 1. Si ottiene FALSE se si verifica un errore.

odbc_field_precision (PHP 4 >= 4.0.0)

Sinonimo di odbc_field_len()

string **odbc_field_precision** (resource id_risultato, int numero_campo) \linebreak

All'interno di un set di righe, referenziate dall'identificativo di risultato ODBC fornito, la funzione **odbc_field_precision()** restituisce la precisione del campo indicato dal numero di campo indicato.

Vedere anche: odbc_field_scale() per ottenere la scala di un numero in virgola mobile.

odbc_field_scale (PHP 4 >= 4.0.0)

Restituisce la scala di un campo

string **odbc_field_scale** (resource id_risultato, int numero_campo) \linebreak

All’interno di un set di righe, referenziate dall’identificativo di risultato ODBC fornito, la funzione odbc_field_precision() restituisce la scala del campo indicato dal numero di campo indicato.

odbc_field_type (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Tipo di dato di campo

string **odbc_field_type** (resource id_risultato, int numero_campo) \linebreak

La funzione **odbc_field_type()** restituisce il tipo di dato SQL del campo indicato dal numero all’interno di un set di righe referenziate dall’identificativo di risultato ODBC passato. La numerazione delle colonne parte da 1.

odbc_foreignkeys (PHP 4 >= 4.0.0)

Restituisce l’elenco delle chiavi esterne per la tabella indicata, oppure la lista delle chiavi esterne in altre tabelle che fanno riferimento alla chiave primaria della tabella indicata.

resource **odbc_foreignkeys** (resource id_connessione, string pk_qualifica, string pk_proprietario, string pk_tabella, string fk_qualifica, string fk_proprietario, string fk_tabella) \linebreak

La funzione **odbc_foreignkeys()** ritorna informazioni sulle chiavi esterne. Restituisce un identificatore di risultato oppure FALSE se si verifica un errore.

Le righe risultanti dall’elaborazione contengono i seguenti campi:

- PKTABLE_QUALIFIER
- PKTABLE_OWNER
- PKTABLE_NAME
- PKCOLUMN_NAME
- FKTABLE_QUALIFIER
- FKTABLE_OWNER
- FKTABLE_NAME
- FKCOLUMN_NAME
- KEY_SEQ
- UPDATE_RULE
- DELETE_RULE
- FK_NAME
- PK_NAME

Se l'argomento *pk_tabella* contiene il nome di una tabella, la funzione **odbc_foreignkeys()** ritorna una serie di righe contenenti i dati della chiave primaria della tabella e di tutte le chiavi esterne che hanno riferimenti a questa.

Se l'argomento *fk_tabella* contiene il nome di una tabella, la funzione **odbc_foreignkeys()** ritorna una serie di righe contenenti i dati delle chiavi esterne della tabella e delle chiavi primarie (di altre tabelle) a cui queste hanno riferimenti.

Se entrambi gli argomenti *pk_tabella* e *fk_tabella* contengono nomi di tabelle, **odbc_foreignkeys()** restituisce le chiavi esterne della tabella specificata in *fk_tabella* che hanno riferimenti alla chiave primaria della tabella indicata in *pk_tabella*. La funzione dovrebbe trovare almeno una chiave.

odbc_free_result (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Libera le risorse associate ad un risultato

bool **odbc_free_result** (resource id_risultato) \linebreak

Restituisce sempre `TRUE`.

La funzione **odbc_free_result()** permette di non utilizzare molta memoria durante l'esecuzione di uno script. Infatti, se si è sicuri di non avere più bisogno dei dati del risultato, si può eseguire **odbc_free_result()**, e la memoria associata a *id_risultato* sarà liberata. Se la funzione non viene utilizzata, le aree di memoria resteranno disponibili per tutta la durata dello script. Al termine verranno liberate in modo automatico.

> **Nota:** Se si ha l'auto-commit disabilitato (vedere odbc_autocommit()) e si esegue **odbc_free_result()** prima di eseguire il commit, tutte le transazioni pendenti saranno annullate,

odbc_gettypeinfo (PHP 4 >= 4.0.0)

Restituisce un identificatore di risultato contenente informazioni sui tipi di dati supportati dalla sorgente di dati.

int **odbc_gettypeinfo** (resource id_connessione [, int tipo_dato]) \linebreak

Recupera informazioni sui tipi di dati supportati dalla sorgente di dati. La funzione restituisce un identificatore di risultato ODBC oppure `FALSE` su errore. L'argomento opzionale *tipo_dato* può essere utilizzato per restringere l'informazione su un singolo tipo.

Le righe risultanti dall'elaborazione contengono i seguenti campi:

- TYPE_NAME
- DATA_TYPE
- PRECISION

- LITERAL_PREFIX
- LITERAL_SUFFIX
- CREATE_PARAMS
- NULLABLE
- CASE_SENSITIVE
- SEARCHABLE
- UNSIGNED_ATTRIBUTE
- MONEY
- AUTO_INCREMENT
- LOCAL_TYPE_NAME
- MINIMUM_SCALE
- MAXIMUM_SCALE

I campi di ordinamento delle righe risultanti sono DATA_TYPE e TYPE_NAME.

odbc_longreadlen (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gestione di colonne LONG

int **odbc_longreadlen** (resource id_risultato, int lunghezza) \linebreak

(tipi di campi ODBC ed SQL coinvolti: LONG, LONGVARBINARY) Tramite l'argomento lunghezza si controlla il numero di byte da ritornare a PHP. Se il campo viene posto a 0, i dati della colonna saranno passati direttamente al client.

> **Nota:** Per la gestione delle colonne di tipo LONGVARBINARY si utilizza anche odbc_binmode().

odbc_next_result (PHP 4 >= 4.0.5)

Verifica se sono disponibili più risultati

bool **odbc_next_result** (resource id_risultato) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

odbc_num_fields (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Numero di colonne in un esito

int **odbc_num_fields** (resource id_risultato) \linebreak

All'interno di un set di righe, referenziate dall'identificativo di risultato ODBC fornito, la funzione **odbc_num_fields()** restituisce il numero di campi (colonne) presenti. La funzione restituisce -1 se si verifica un errore. L'argomento fornito è un identificatore di esito restituito dalla funzione odbc_exec().

odbc_num_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Numero di righe in un risultato

int **odbc_num_rows** (resource id_risultato) \linebreak

odbc_num_rows() ritorna il numero di record presenti in un risultato ODBC. La funzione ritorna -1 se si verifica un errore. Per le clausole INSERT, UPDATE e DELETE, **odbc_num_rows()** ritorna il numero di righe coinvolte. Nella clausola SELECT questo può essere il numero di righe disponibili.

Nota: Con diversi driver, la funzione **odbc_num_rows()**, utilizzata con lo scopo di determinare il numero di righe dopo una SELECT, restituisce -1.

odbc_pconnect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Apre una connessione persistente verso un database

int **odbc_pconnect** (string dsn, string utente, string password [, int tipo_cursore]) \linebreak

Restituisce un identificatore di connessione ODBC oppure 0 (FALSE) su errore. Questa funzione è molto simile a odbc_connect(), eccetto che la connessione non viene realmente chiusa quando lo script finisce. Successive richieste di connessione che utilizzino la stessa combinazione di *dsn*, *utente*, *password* (eseguite sia utilizzando odbc_connect(), sia utilizzando **odbc_pconnect()**) possono riutilizzare la connessione.

> **Nota:** Le connessioni persistenti non hanno effetti se PHP viene utilizzato come programma CGI.

Per informazioni sul campo opzionale tipo_cursore, vedere la funzione odbc_connect(). Per maggiori dettagli sulle connessioni persistenti, fare riferimento alla FAQ di PHP.

odbc_prepare (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Predispone un'espressione all'esecuzione

resource **odbc_prepare** (resource id_connessione, string testo_query) \linebreak

La funzione restituisce FALSE su errore.

Restituisce un identificativo di risultato ODBC se l'espressione SQL viene predisposta correttamente. L'identificativo restituito può essere utilizzato successivamente per eseguire l'espressione utilizzando la funzione odbc_execute().

odbc_primarykeys (PHP 4 >= 4.0.0)

Restituisce un identificatore di risultato che può essere utilizzato per ricavare il nome della colonna che contiene la chiave primaria della tabella.

resource **odbc_primarykeys** (resource id_connessione, string qualifica, string proprietario, string tabella) \linebreak

Restituisce il nome della colonna che contiene la chiave primaria per la tabella. La funzione ritorna un identificatore di risultato ODBC oppure FALSE se si verifica un errore.

Le righe risultanti dall'elaborazione contengono i seguenti campi:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- COLUMN_NAME
- KEY_SEQ
- PK_NAME

odbc_procedurecolumns (PHP 4 >= 4.0.0)

Recupera informazioni sui parametri delle procedure.

resource **odbc_procedurecolumns** (resource id_connessione [, string qualifica [, string proprietario [, string procedura [, string colonna]]]]) \linebreak

La funzione ritorna la lista dei parametri di input e di output e anche delle colonne che concorrono al determinazione del risultato per le procedure indicate. Viene restituito un identificatore di risultato oppure FALSE se si è un errore.

Le righe risultanti dall'elaborazione contengono i seguenti campi:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER

- PROCEDURE_NAME
- COLUMN_NAME
- COLUMN_TYPE
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- RADIX
- NULLABLE
- REMARKS

I campi di ordinamento delle righe risultanti sono PROCEDURE_QUALIFIER, PROCEDURE_OWNER, PROCEDURE_NAME e COLUMN_TYPE.

Gli argomenti *proprietario*, *procedura* e *colonna* accettano dei criteri di ricerca ('%' per indicare zero o più caratteri e '_' per indicare un singolo carattere).

odbc_procedures (PHP 4 >= 4.0.0)

Restituisce l'elenco delle procedure memorizzate in una specifica sorgente di dati. La funzione ritorna un identificatore di risultato che punta alle informazioni reperite.

resource **odbc_procedures** (resource id_connessione [, string qualifica [, string proprietario [, string nome]]]) \linebreak

Si ottiene l'elenco di tutte le procedure presenti nei limiti richiesti. La funzione restituisce un identificatore di risultato, oppure `FALSE` su errore.

Le righe risultanti dall'elaborazione contengono i seguenti campi:

- PROCEDURE_QUALIFIER
- PROCEDURE_OWNER
- PROCEDURE_NAME
- NUM_INPUT_PARAMS
- NUM_OUTPUT_PARAMS
- NUM_RESULT_SETS
- REMARKS
- PROCEDURE_TYPE

Gli argomenti *proprietario* e *nome* accettano dei criteri di ricerca ('%' per indicare zero o più caratteri e '_' per indicare un singolo carattere).

odbc_result (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce il contenuto dei campi

string **odbc_result** (resource id_risultato, mixed campo) \linebreak

Restituisce il contenuto dei campi.

Il parametro *campo* può essere sia un intero indicante il numero di colonna del campo desiderato; sia una stringa contenente il nome del campo. Ad esempio:

```
$item_3 = odbc_result ($Query_ID, 3);
$item_val = odbc_result ($Query_ID, "val");
```

Nel primo caso l'esecuzione di **odbc_result()** restituisce il valore del terzo campo del record corrente della query. Nel secondo, la funzione **odbc_result()** restituisce il valore del campo il cui nome è "val", sempre utilizzando i dati dal record corrente. Si ha un errore qualora il numero di colonna fornito sia minore di 1 oppure sia superiore al numero delle colonne (o campi) presenti nel record corrente. Analogamente, si ottiene un errore se il nome del campo richiesto non sia presente nella tabella/e oggetto della ricerca.

L'indice dei campi parte da 1. Per quanto riguarda la gestione dei campi di tipo binario o long fare riferimento a odbc_binmode() e a odbc_longreadlen().

odbc_result_all (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Visualizza il risultato in una tabella HTML

int **odbc_result_all** (resource id_esito [, string formato]) \linebreak

Restituisce il numero di righe elaborate, oppure `FALSE` se si verifica un errore.

Dato un identificatore di risultato restituito da odbc_exec(), la funzione **odbc_result_all()** visualizza tutti i record ottenuti in una di tabella in formato HTML. Utilizzando il parametro opzionale *formato*, è possibile fornire informazioni addizionali sulla formattazione della tabella.

odbc_rollback (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Annulla una transazione

int **odbc_rollback** (resource id_connessione) \linebreak

Annulla tutte le operazioni pendenti sulla connessione indicata da *id_connessione*. Se ha successo restituisce `TRUE`, altrimenti `FALSE`.

odbc_setoption (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Settaggio dei parametri ODBC. Restituisce `FALSE` se si verifica un errore, altrimenti `TRUE`.

int **odbc_setoption** (resource identificativo, int funzione, int opzione, int parametro) \linebreak

Questa funzione permette di manipolare i parametri ODBC per la connessione o il risultato di una query indicati. La funzione è stata sviluppata per permettere di aggirare dei problemi emersi in alcuni driver ODBC. Pertanto si dovrebbe utilizzare questa funzione soltanto se si è dei programmatori e si conoscono gli effetti generati dalle varie opzioni. Dato che ogni singola versione di driver ODBC supporta differenti parametri, occorre avere a disposizione un buon manuale del driver per avere esposti tutti i differenti settaggi che possono essere utilizzati.

Poiché i parametri possono variare in base al driver ODBC, è fortemente sconsigliato l'uso di questa funzione in script resi pubblici. Inoltre, alcune opzioni di ODBC non sono gestibili da questa funzione, dato che devono essere specificate prima di stabilire la connessione o prima della preparazione della query. Tuttavia, se per un particolare lavoro permette al PHP di funzionare, può evitare il ricorso a prodotti commerciali.

Il campo *`identificativo`* indica la connessione o l'esito su cui si varia il settaggio. Per la funzione SQLSetConnectOption(), questo indica l'identificativo di connessione, per SQLSetStmtOption(), indica l'identificativo del risultato.

Il campo *`funzione`* indica quale funzione ODBC utilizzare. Dovrebbe essere valorizzato a 1 per SQLSetConnectOption() e a 2 per SQLSetStmtOption().

Il parametro *`opzione`* indica l'opzione da settare.

Il campo *`parametro`* indica il valore per l'*`opzione`* richiesta.

Esempio 1. Esempi di utilizzo

```
// 1. Il valore 102 per il campo opzione in SQLSetConnectOption() indica SQL_AUTOCOMMIT.
//    Il valore 1 per SQL_AUTOCOMMIT è SQL_AUTOCOMMIT_ON.
//    Pertanto questo esempio ha il medesimo effetto di:
//    odbc_autocommit($conn, true);

odbc_setoption ($conn, 1, 102, 1);

// 2. Il valore 0 per il campo opzione in SQLSetStmtOption() indica SQL_QUERY_TIMEOUT.
//    In questo esempio si setta il timeout di una query a 30 secondi.

$result = odbc_prepare ($conn, $sql);
odbc_setoption ($result, 2, 0, 30);
odbc_execute ($result);
```

odbc_specialcolumns (PHP 4 >= 4.0.0)

Restituisce sia il set di colonne che identificano in modo univoco una riga nella tabella, sia colonne che sono automaticamente aggiornate quando un qualsiasi campo della riga viene aggiornato da una transazione.

resource **odbc_specialcolumns** (resource id_connessione, int tipo, string qualifica, string proprietario, string tabella, int visibilità, int nullable) \linebreak

Quando l'argomento tipo è impostato a SQL_BEST_ROWID, **odbc_specialcolumns()** restituisce la colonna o le colonne che identificano in modo univoco ciascuna riga nella tabella.

Quando l'argomento tipo è impostato a SQL_ROWVER, **odbc_specialcolumns()** restituisce la colonna o il set di colonne ottimali, attraverso cui, ottenendo i valori da dette colonne, è possibile identificare in modo univoco ciascun record della tabella indicata.

La funzione restituisce un identificatore di risultato ODBC, oppure `FALSE` su errore.

Le righe risultanti dall'elaborazione contengono i seguenti campi:

- SCOPE
- COLUMN_NAME
- DATA_TYPE
- TYPE_NAME
- PRECISION
- LENGTH
- SCALE
- PSEUDO_COLUMN

Le righe del risultato sono ordinate in base alla colonna SCOPE.

odbc_statistics (PHP 4 >= 4.0.0)

Recupera informazioni statistiche sulla tabella

resource **odbc_statistics** (resource id_connessione, string qualifica, string proprietario, string nome_tabella, int unico, int precisione) \linebreak

Si ottengono informazioni statistiche sulla tabella e i propri indici. La funzione restituisce un identificatore di risultato ODBC, oppure `FALSE` su errore.

Le righe risultanti dall'elaborazione contengono i seguenti campi:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- NON_UNIQUE

- INDEX_QUALIFIER
- INDEX_NAME
- TYPE
- SEQ_IN_INDEX
- COLUMN_NAME
- COLLATION
- CARDINALITY
- PAGES
- FILTER_CONDITION

I campi di ordinamento delle righe risultanti sono NON_UNIQUE, TYPE, INDEX_QUALIFIER, INDEX_NAME e SEQ_IN_INDEX.

odbc_tableprivileges (PHP 4 >= 4.0.0)

Elenca le tabelle ed i privilegi ad esse associati.

int **odbc_tableprivileges** (resource id_connessione [, string qualifica [, string proprietario [, string nome]]]) \linebreak

Elenca le tabelle presenti nei limiti richiesti e, per ciascuna di queste, ne fornisce i privilegi. La funzione ritorna un identificatore di risultato ODBC, oppure `FALSE` su errore.

Le righe risultanti dall'elaborazione hanno i seguenti campi:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- GRANTOR
- GRANTEE
- PRIVILEGE
- IS_GRANTABLE

I campi di ordinamento delle righe risultanti sono TABLE_QUALIFIER, TABLE_OWNER e TABLE_NAME.

Gli argomenti *`proprietario`* e *`nome`* accettano dei criteri di ricerca ('%' per indicare zero o più caratteri e '_' per indicare un singolo carattere).

odbc_tables (PHP 3>= 3.0.17, PHP 4 >= 4.0.0)

Restituisce l'elenco delle tabelle presenti in una specifica sorgente di dati. Restituisce l'identificatore di risultato in cui vi sono le informazioni.

int **odbc_tables** (resource Id_connessione [, string qualifica [, string proprietario [, string nome [, string tipo]]]]) \linebreak

La funzione elenca tutte le tabelle presenti nei limiti richiesti. Restituisce un identificatore di risultato oppure `FALSE` se si verifica un errore.

Le righe risultanti hanno i seguenti campi:

- TABLE_QUALIFIER
- TABLE_OWNER
- TABLE_NAME
- TABLE_TYPE
- REMARKS

I campi di ordinamento delle righe risultanti sono TABLE_TYPE, TABLE_QUALIFIER, TABLE_OWNER e TABLE_NAME.

Gli argomenti *`proprietario`* e *`nome`* accettano dei criteri di ricerca ('%' per indicare zero o più caratteri e '_' per indicare un singolo carattere).

Per supportare l'enumerazione delle qualifiche, dei proprietari e dei tipi tabelle, è stata predisposta la seguente semantica per i campi *`qualifica`*, *`proprietario`*, *`nome`*, e *`tipo`*:

- Se l'argomento *`qualifica`* è valorizzato con il carattere percento (%) e i parametri *`proprietario`* e *`nome`* sono delle stringhe vuote, il risultato sarà un set di righe contenente la lista delle qualifiche previste per la sorgente di dati. (Tutte le colonne tranne TABLE_QUALIFIER conterranno NULL.)
- Se l'argomento *`proprietario`* è valorizzato con il carattere percento (%) e i parametri *`qualifica`* e *`nome`* sono delle stringhe vuote, il risultato sarà un set di righe contenente la lista dei proprietari previsti per la sorgente di dati. (Tutte le colonne tranne TABLE_OWNER conterranno NULL.)
- Se l'argomento *`tipo`* è valorizzato con il carattere percento (%) e i parametri *`qualifica`*, *`proprietario`* e *`nome`* sono delle stringhe vuote, il risultato sarà un set di righe contenente la lista dei tipi di tabella previsti per la sorgente di dati. (Tutte le colonne tranne TABLE_TYPE conterranno NULL.)

Se l'argomento *`tipo`* non è una stinga vuota, deve contenere l'elenco dei tipi interessati separati dalla virgola; ogni singolo valore può essere, o meno, racchiuso tra apici singoli ('). Ad esempio: "'TABLE','VIEW'" o "TABLE, VIEW" sono valori validi. Se la sorgente di dati non supporta alcuni dei tipi di tabelle specificati, per questi, la funzione **odbc_tables()** non riporta alcuna informazione.

Vedere inoltre odbc_tableprivileges() per ottenere i privilegi associati alla tabella.

LXIX. Funzioni Oracle 8

Queste funzioni permettono di accedere ai database Oracle8 e Oracle7. Usano la Call-Interface di Oracle8 (OCI8). Occorre avere installate le librerie client di Oracle8 per utilizzare questa estensione.

Questa estensione è più flessibile della estensione standard di Oracle. Supporta il binding di variabili PHP locali e globali ai segnaposto Oracle, ha pieno supporto di LOB, FILE e ROWID e permette di utilizzare variabili di definizione personalizzabili.

Prima di usare questa estensione, occorre sicerarsi di aver impostato le variabili d'ambiente per l'utente Oracle, come pure per l'utente del server web. Le variabili che potrebbero necessitare l'impostazione sono le seguenti:

- ORACLE_HOME
- ORACLE_SID
- LD_PRELOAD
- LD_LIBRARY_PATH
- NLS_LANG
- ORA_NLS33

Dopo aver impostato le variabili d'ambiente per l'utente del server web, occorre sicerarsi di aver aggiunto anche l'utente stesso (nobody, www) al gruppo oracle.

Se il server web non parte o va in blocco: Controllare che apache sia linkato con la libreria pthread:

```
# ldd /www/apache/bin/httpd
    libpthread.so.0 => /lib/libpthread.so.0 (0x4001c000)
    libm.so.6 => /lib/libm.so.6 (0x4002f000)
    libcrypt.so.1 => /lib/libcrypt.so.1 (0x4004c000)
    libdl.so.2 => /lib/libdl.so.2 (0x4007a000)
    libc.so.6 => /lib/libc.so.6 (0x4007e000)
    /lib/ld-linux.so.2 => /lib/ld-linux.so.2 (0x40000000)
```

Se la libpthread non compare nell'elenco, occorre reinstallare Apache:

```
# cd /usr/src/apache_1.3.xx
# make clean
# LIBS=-lpthread ./config.status
# make
# make install
```

Esempio 1. Trucchi OCI

```
<?php
// by sergo@bacup.ru

// Usare l'opzione OCI_DEFAULT nel comando execute per ritardare l'esicuzione
OCIExecute($stmt, OCI_DEFAULT);

// per ricevere i dati utilizzare (dopo il fetch):

$result = OCIResult($stmt, $n);
if (is_object ($result)) $result = $result->load();

// come comandi  INSERT o UPDATE usare:

$sql = "insert into table (field1, field2) values (field1 = 'value',
 field2 = empty_clob()) returning field2 into :field2";
OCIParse($conn, $sql);
$clob = OCINewDescriptor($conn, OCI_D_LOB);
OCIBindByName ($stmt, ":field2", &$clob, -1, OCI_B_CLOB);
OCIExecute($stmt, OCI_DEFAULT);
$clob->save ("some text");
OCICommit($conn);

?>
```

You can easily access stored procedures in the same way as you would from the commands line.

Esempio 2. Using Stored Procedures

```
<?php
// by webmaster@remoterealty.com
$sth = OCIParse ( $dbh, "begin sp_newaddress( :address_id, '$firstname',
 '$lastname', '$company', '$address1', '$address2', '$city', '$state',
 '$postalcode', '$country', :error_code );end;" );

// Questo codice richiama la stord procedure sp_newaddress, dove :address_id è
// una variabile in/out e :error_code è una variabile out.
// Quindi si effettua il binding:

   OCIBindByName ( $sth, ":address_id", $addr_id, 10 );
   OCIBindByName ( $sth, ":error_code", $errorcode, 10 );
   OCIExecute ( $sth );

?>
```

OCIBindByName (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Lega una variabile PHP ad un segnaposto Oracle

int **OCIBindByName** (int stmt, string ph_name, mixed & variable, int length [, int type]) \linebreak

OCIBindByName() collega la variabile PHP *variable* al segnaposto Oracle *ph_name*. L'utilizzo in modalità input o output sarà determinato a run-time, e lo spazio di memoria necessario sarà allocato. Il parametro *length* imposta la lunghezza massima del collegamento. Se si imposta *length* a -1 **OCIBindByName()** userà l'attuale lunghezza di *variable* per impostare la lunghezza massima.

Se si deve collegare un Datatype astratto (LOB/ROWID/BFILE) occorre innanzitutto allocarlo usando la funzione **OCINewDescriptor()**. Il parametro *length* non è usato con i Datatypes astratti e dovrebbe essere impostato a -1. La veriabile *type* informa oracle sul tipo di descrittore che si vuole usare. I valori possibili sono: OCI_B_FILE (Binary-File), OCI_B_CFILE (Character-File), OCI_B_CLOB (Character-LOB), OCI_B_BLOB (Binary-LOB) e OCI_B_ROWID (ROWID).

Esempio 1. OCIDefineByName

```
<?php
/* OCIBindByPos example thies@thieso.net (980221)
  inserisce 3 tuple in emp, e usa ROWID per aggiornare le
  tuple subito dopo l'inserimento.
*/

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"insert into emp (empno, ename) ".
   "values (:empno,:ename) ".
   "returning ROWID into :rid");

$data = array(1111 => "Larry", 2222 => "Bill", 3333 => "Jim");

$rowid = OCINewDescriptor($conn,OCI_D_ROWID);

OCIBindByName($stmt,":empno",&$empno,32);
OCIBindByName($stmt,":ename",&$ename,32);
OCIBindByName($stmt,":rid",&$rowid,-1,OCI_B_ROWID);

$update = OCIParse($conn,"update emp set sal = :sal where ROWID = :rid");
OCIBindByName($update,":rid",&$rowid,-1,OCI_B_ROWID);
OCIBindByName($update,":sal",&$sal,32);

$sal = 10000;

while (list($empno,$ename) = each($data)) {
OCIExecute($stmt);
OCIExecute($update);
}

$rowid->free();

OCIFreeStatement($update);
OCIFreeStatement($stmt);
```

```
$stmt = OCIParse($conn,"select * from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
while (OCIFetchInto($stmt,&$arr,OCI_ASSOC)) {
var_dump($arr);
}
OCIFreeStatement($stmt);

/* delete our "junk" from the emp table.... */
$stmt = OCIParse($conn,"delete from emp where empno in (1111,2222,3333)");
OCIExecute($stmt);
OCIFreeStatement($stmt);

OCILogoff($conn);
?>
```

Attenzione

It is a bad idea to use magic quotes and **OciBindByName()** simultaneously as no quoting is needed on quoted variables and any quotes magically applied will be written into your database as **OciBindByName()** is not able to distinguish magically added quotings from those added by intention.

OCICancel (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Interrompe la lettura del cursore

int **OCICancel** (int stmt) \linebreak

Se non si vogliono leggere altri dati da un cursore, chiamare **OCICancel()**.

OCICollAppend (PHP 4 >= 4.0.6)

Coming soon

string **OCICollAppend** (object collection, object object) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCICollAssign (PHP 4 >= 4.0.6)

Coming soon

string **OCICollAssign** (object collection, object object) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCICollAssignElem (PHP 4 >= 4.0.6)

Coming soon

string **OCICollAssignElem** (object collection, string ndx, string val) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCICollGetElem (PHP 4 >= 4.0.6)

Coming soon

string **OCICollGetElem** (object collection, string ndx) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCICollMax (PHP 4 >= 4.0.6)

Coming soon

string **OCICollMax** (object collection) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCICollSize (PHP 4 >= 4.0.6)

Coming soon

string **OCICollSize** (object collection) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCICollTrim (PHP 4 >= 4.0.6)

Coming soon

string **OCICollTrim** (object collection, int num) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCIColumnIsNULL (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Verifica se un campo di risultato è `NULL`

int **OCIColumnIsNULL** (int stmt, mixed column) \linebreak

OCIColumnIsNULL() restituisce `TRUE` se il campo *`col`* nel risultato dell'istruzione *`stmt`* è `NULL`. Si può usare il numero del campo (primo campo=1) o il nome del campo per il parametro *`col`*.

OCIColumnName (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Restituisce il nome di un campo

string **OCIColumnName** (int stmt, int col) \linebreak

OCIColumnName() restituisce il nome del campo corrispondente alla posizione (1 = primo campo) specificata.

Esempio 1. OCIColumnName

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vedere anche **OCINumCols()**, **OCIColumnType()**, e **OCIColumnSize()**.

OCIColumnPrecision (PHP 4 >= 4.0.0)

Coming soon

int **OCIColumnPrecision** (int stmt, int col) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCIColumnScale (PHP 4 >= 4.0.0)

Coming soon

int **OCIColumnScale** (int stmt, int col) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCIColumnSize (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Restituisce la dimensione del campo

int **OCIColumnSize** (int stmt, mixed column) \linebreak

OCIColumnSize() restituisce la dimensione del campo come riportata da Oracle. Si può usare il numero del campo (primo campo=1) o il nome del campo per il parametro *col*.

Esempio 1. OCIColumnSize

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
```

```
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vedere anche **OCINumCols()**, **OCIColumnName()**, e **OCIColumnSize()**.

OCIColumnType (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Restituisce il tipo di dati di un campo

mixed **OCIColumnType** (int stmt, int col) \linebreak

OCIColumnType() restituisce il tipo del campo corrispondente alla posizione (1 = primo campo) specificata.

Esempio 1. OCIColumnType

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    print "<TABLE BORDER=\"1\">";
    print "<TR>";
    print "<TH>Name</TH>";
    print "<TH>Type</TH>";
    print "<TH>Length</TH>";
    print "</TR>";
    $ncols = OCINumCols($stmt);
    for ( $i = 1; $i <= $ncols; $i++ ) {
        $column_name  = OCIColumnName($stmt,$i);
        $column_type  = OCIColumnType($stmt,$i);
        $column_size  = OCIColumnSize($stmt,$i);
        print "<TR>";
        print "<TD>$column_name</TD>";
        print "<TD>$column_type</TD>";
        print "<TD>$column_size</TD>";
        print "</TR>";
    }
    print "</TABLE>\n";
    OCIFreeStatement($stmt);
```

```
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

Vedere anche **OCINumCols()**, **OCIColumnName()**, e **OCIColumnSize()**.

OCIColumnTypeRaw (PHP 4 >= 4.0.0)

Coming soon

mixed **OCIColumnTypeRaw** (int stmt, int col) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCICommit (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Esegue le transazioni in sospeso

int **OCICommit** (int connection) \linebreak

OCICommit() esegue tutte le transazioni in sospeso sulla connessione Oracle *`connection`*.

OCIDefineByName (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Utilizza una variabile PHP per la fase di definizione in un comando SELECT

int **OCIDefineByName** (int stmt, string Column-Name, mixed variable [, int type]) \linebreak

OCIDefineByName() aggancia le variabili PHP ai campi SQL. Attenzione: Oracle usa nomi di colonna MAIUSCOLI, mentre nella SELECT si possono anche scrivere minuscoli. **OCIDefineByName()** vuole il parametro *`Column-Name`* in caratteri maiuscoli. Se si definisce una variabile che non esiste nel comando SELECT, non viene dato alcun errore!

Se occorre definire un tipo di dati astratto (LOB/ROWID/BFILE) lo si deve prima allocare usando la funzione **OCINewDescriptor()**. Vedere anche la funzione **OCIBindByName()**.

Esempio 1. OCIDefineByName

```
<?php
/* OCIDefineByName example thies@thieso.net (980219) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select empno, ename from emp");

/* il define DEVE essere eseguito PRIMA di ociexecute! */

OCIDefineByName($stmt,"EMPNO",$empno);
OCIDefineByName($stmt,"ENAME",$ename);

OCIExecute($stmt);

while (OCIFetch($stmt)) {
    echo "empno:".$empno."\n";
    echo "ename:".$ename."\n";
}

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIError (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Restituisce l'ultimo errore di stmt|conn|global

array **OCIError** ([int stmt|conn|global]) \linebreak

OCIError() restituisce l'ultimo errore. Se il parametro opzionale *stmt|conn|global* non è specificato, viene restituito l'ultimo errore generato. Se non ci sono errori, **OCIError()** restituisce FALSE. **OCIError()** restituisce l'errore in un array associativo. In questo array, *code* dà il codice d'errora oracle e *message* dà la stringa d'errore.

OCIExecute (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Esegue un comando SQL

int **OCIExecute** (int statement [, int mode]) \linebreak

OCIExecute() esegue un comando precedentemente analizzato. (vedere **OCIParse()**). Il parametro opzionale *mode* permette di specificare la modalità di esecuzione (il default è OCI_COMMIT_ON_SUCCESS). Se non si desidera che i comandi eseguano un commit automatico, usare OCI_DEFAULT nella variabile *mode*.

Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

OCIFetch (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Estrae la prossima tupla opnendola nel buffer di risultato.

int **OCIFetch** (int statement) \linebreak

OCIFetch() estrae la prossima tupla (nelle istruzioni SELECT) ponendola nel buffer interno di risultato.

OCIFetchInto (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Estrae la prossima tupla ponendola in un array

int **OCIFetchInto** (int stmt, array & result [, int mode]) \linebreak

OCIFetchInto() estrae la prossima tupla (nelle istruzioni SELECT) ponendola nell'array *`result`*. **OCIFetchInto()** sovrascriverà il preceente contenuto di *`result`*. Di default *`result`* conterrà un array (primo indice = 1) di tutti i campi che non sono `NULL`.

Il parametro *`mode`* permette di cambiare il comportamento predefinito. E' possibile specificare più di un'opzione sommandole (es. OCI_ASSOC+OCI_RETURN_NULLS). Le opzioni valide sono:

`OCI_ASSOC` Restituisce un array associativo.
`OCI_NUM` Restituisce un array indicizzato (primo indice = 1). (DEFAULT)
`OCI_RETURN_NULLS` Restituisce anche i campi `NULL`.
`OCI_RETURN_LOBS` Restituisce il valore di un LOB invece del descrittore.

OCIFetchStatement (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Estrae tutte le tuple in un array

int **OCIFetchStatement** (int stmt, array & variable) \linebreak

OCIFetchStatement() estrae tutte le tuple da un risultato ponendole in un array definito dall'utente. **OCIFetchStatement()** restituisce il numero di tuple estratte.

Esempio 1. OCIFetchStatement

```
<?php
/* OCIFetchStatement example mbritton@verinet.com (990624) */

$conn = OCILogon("scott","tiger");

$stmt = OCIParse($conn,"select * from emp");

OCIExecute($stmt);

$nrows = OCIFetchStatement($stmt,$results);
if ( $nrows > 0 ) {
   print "<TABLE BORDER=\"1\">\n";
```

```
   print "<TR>\n";
   while ( list( $key, $val ) = each( $results ) ) {
      print "<TH>$key</TH>\n";
   }
   print "</TR>\n";

   for ( $i = 0; $i < $nrows; $i++ ) {
      reset($results);
      print "<TR>\n";
      while ( $column = each($results) ) {
         $data = $column['value'];
         print "<TD>$data[$i]</TD>\n";
      }
      print "</TR>\n";
   }
   print "</TABLE>\n";
} else {
   echo "No data found<BR>\n";
}
print "$nrows Records Selected<BR>\n";

OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCIFreeCollection (PHP 4 >= 4.1.0)

Coming soon

string **OCIFreeCollection** (object lob) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCIFreeCursor (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Libera tutte le risorse associate ad un cursore

int **OCIFreeCursor** (int stmt) \linebreak

OCIFreeCursor() restituisce `TRUE` in caso di successo, `FALSE` altrimenti.

OCIFreeDesc (PHP 4 >= 4.0.0)

Cancella un descrittore di oggetto binario (LOB)

int **OCIFreeDesc** (object lob) \linebreak

OCIFreeDesc() restituisce `TRUE` in caso di successo, `FALSE` altrimenti.

OCIFreeStatement (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Libera tutte le risorse associate ad un'istruzione

int **OCIFreeStatement** (int stmt) \linebreak

OCIFreeStatement() restituisce `TRUE` in caso di successo, `FALSE` altrimenti.

OCIInternalDebug (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Abilita o disabilita la visualizzazione del debug interno.

void **OCIInternalDebug** (int onoff) \linebreak

OCIInternalDebug() abilita la visualizzazione del debug interno. Impostare *`onoff`* a 0 per spegnere il debug, 1 per accenderlo.

OCILoadLob (PHP 4 >= 4.0.0)

Coming soon

string **OCILoadLob** (object lob) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCILogOff (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Disconnette da Oracle

int **OCILogOff** (int connection) \linebreak

OCILogOff() chiude una connessione Oracle.

OCILogon (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Stabilisce una connessione a Oracle

int **OCILogon** (string username, string password [, string db]) \linebreak

OCILogon() restituisce un identificatore di connessione necessario per la maggior parte delle altre chiamate OCI. Il terzo parametro opzionale pu‘ contenere il nome della istanza Oracle locale o il nome della voce in tnsnames.ora a cui ci si vuole connettere. Se il terzo parametro opzionale non è specificato, PHP usa la variabile d’ambiente ORACLE_SID (istanza di Oracle) o TWO_TASK (in tnsnames.ora) per determinare a quale database collegarsi.

Le connessioni sono condivise a livello di pagina quando si usa **OCILogon()**. Questo significa che i commit e i rollback avvengono su tutte le transazioni aperte nella pagina, anche se sono state create connessioni multiple.

Questo esempio dimostra come le connessioni sono condivise.

Esempio 1. OCILogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocilogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." committed\n\n";
}
```

```
function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."----done\n\n";
}

create_table($c1);
insert_data($c1);   // Insert a row using c1
insert_data($c2);   // Insert a row using c2

select_data($c1);   // Results of both inserts are returned
select_data($c2);

rollback($c1);      // Rollback using c1

select_data($c1);   // Both inserts have been rolled back
select_data($c2);

insert_data($c2);   // Insert a row using c2
commit($c2);        // commit using c2

select_data($c1);   // result of c2 insert is returned

delete_data($c1);   // delete all rows in table using c1
select_data($c1);   // no rows returned
select_data($c2);   // no rows returned
commit($c1);        // commit using c1

select_data($c1);   // no rows returned
select_data($c2);   // no rows returned

drop_table($c1);
print "</PRE></HTML>";
?>
```

Vedere anche **OCIPLogon()** e **OCINLogon()**.

OCINewCollection (PHP 4 >= 4.0.6)

Coming soon

string **OCINewCollection** (int conn, string tdo [, string shema]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCINewCursor (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Restituisce un nuovo cursore (Statement-Handle)

int **OCINewCursor** (int conn) \linebreak

OCINewCursor() alloca un nuovo identificatore di istruzione sulla connessione specificata.

Esempio 1. Using a REF CURSOR from a stored procedure

```
<?php
// suppose your stored procedure info.output returns a ref cursor in :data

$conn = OCILogon("scott","tiger");
$curs = OCINewCursor($conn);
$stmt = OCIParse($conn,"begin info.output(:data); end;");

ocibindbyname($stmt,"data",&$curs,-1,OCI_B_CURSOR);
ociexecute($stmt);
ociexecute($curs);

while (OCIFetchInto($curs,&$data)) {
    var_dump($data);
}

OCIFreeStatement($stmt);
OCIFreeCursor($curs);
OCILogoff($conn);
?>
```

Esempio 2. Using a REF CURSOR in a select statement

```
<?php
print "<HTML><BODY>";
$conn = OCILogon("scott","tiger");
$count_cursor = "CURSOR(select count(empno) num_emps from emp " .
                "where emp.deptno = dept.deptno) as EMPCNT from dept";
$stmt = OCIParse($conn,"select deptno,dname,$count_cursor");

ociexecute($stmt);
```

```
print "<TABLE BORDER=\"1\">";
print "<TR>";
print "<TH>DEPT NAME</TH>";
print "<TH>DEPT #</TH>";
print "<TH># EMPLOYEES</TH>";
print "</TR>";

while (OCIFetchInto($stmt,&amp;$data,OCI_ASSOC)) {
    print "<TR>";
    $dname  = $data["DNAME"];
    $deptno = $data["DEPTNO"];
    print "<TD>$dname</TD>";
    print "<TD>$deptno</TD>";
    ociexecute($data["EMPCNT"]);
    while (OCIFetchInto($data["EMPCNT"],&$subdata,OCI_ASSOC)) {
        $num_emps = $subdata["NUM_EMPS"];
        print  "<TD>$num_emps</TD>";
    }
    print "</TR>";
}
print "</TABLE>";
print "</BODY></HTML>";
OCIFreeStatement($stmt);
OCILogoff($conn);
?>
```

OCINewDescriptor (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Inizializza un nuovo descrittore LOB/FILE vuoto

string **OCINewDescriptor** (int connection [, int type]) \linebreak

OCINewDescriptor() alloca memoria per accogliere descrittori o locatori LOB. I valori validi per il parametro *type* sono OCI_D_FILE, OCI_D_LOB, OCI_D_ROWID. Per i descrittori LOB, i metodi load, save, e savefile sono associati al descrittore, per i BFILE esiste solo il metodo load. Vedere i suggerimenti nel secondo esempio.

Esempio 1. OCINewDescriptor

```
<?php
    /* Questo codice deve essere richiamato da un form HTML.
     * Richiede che $user, $password, $table, $where, e $commitsize
     * siano passati dalla form.  Il codice quindi cancella
     * le tuple selezionate usando ROWID ed esegue un commit ogni
     * $commitsize righe. (Usare con attenzione, non si può fare rollback)
     */
    $conn = OCILogon($user, $password);
    $stmt = OCIParse($conn,"select rowid from $table $where");
    $rowid = OCINewDescriptor($conn,OCI_D_ROWID);
```

```
    OCIDefineByName($stmt,"ROWID",&$rowid);
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
       $nrows = OCIRowCount($stmt);
       $delete = OCIParse($conn,"delete from $table where ROWID = :rid");
       OCIBindByName($delete,":rid",&$rowid,-1,OCI_B_ROWID);
       OCIExecute($delete);
       print "$nrows\n";
       if ( ($nrows % $commitsize) == 0 ) {
           OCICommit($conn);
       }
    }
    $nrows = OCIRowCount($stmt);
    print "$nrows deleted...\n";
    OCIFreeStatement($stmt);
    OCILogoff($conn);
?>
```

```
<?php
    /* Questo codice dimostra l’upload di file verso campi LOB.
     * Il form usato per questo esempio è del tipo seguente:
     * <form action="upload.php" method="post" enctype="multipart/form-data">
     * <input type="file" name="lob_upload">
     * ...
     */
  if(!isset($lob_upload) || $lob_upload == ’none’){
?>
<form action="upload.php" method="post" enctype="multipart/form-data">
Upload file: <input type="file" name="lob_upload"><br>
<input type="submit" value="Upload"> - <input type="reset">
</form>
<?php
} else {

     // $lob_upload contiene il nome del file temporaneo

     // vedere anche la sezione delle funzionalita’ di upload dei file,
     // se si vogliono usare gli upload sicuri

     $conn = OCILogon($user, $password);
     $lob = OCINewDescriptor($conn, OCI_D_LOB);
     $stmt = OCIParse($conn,"insert into $table (id, the_blob)
               values(my_seq.NEXTVAL, EMPTY_BLOB()) returning the_blob into :the_blob");
     OCIBindByName($stmt, ’:the_blob’, &$lob, -1, OCI_B_BLOB);
     OCIExecute($stmt, OCI_DEFAULT);
     if($lob->savefile($lob_upload)){
        OCICommit($conn);
        echo "Blob successfully uploaded\n";
     }else{
        echo "Couldn’t upload Blob\n";
     }
     OCIFreeDesc($lob);
     OCIFreeStatement($stmt);
     OCILogoff($conn);
  }
```

```
?>
```

Esempio 2. OCINewDescriptor

```
<?php
    /* Chiamatai di una stored procedure PL/SQLs che contiene clobs come parametri
    * di input (PHP 4 >= 4.0.6).
    * La signature della stored prodedure PL/SQL d’esempio è:
     *
     * PROCEDURE save_data
     *   Argument Name                  Type                   In/Out Default?
     *   ------------------------------ ---------------------- ------ ----
----
     *   KEY                            NUMBER(38)             IN
     *   DATA                           CLOB                   IN
     *
     */

    $conn = OCILogon($user, $password);
    $stmt = OCIParse($conn, "begin save_data(:key, :data); end;");
    $clob = OCINewDescriptor($conn, OCI_D_LOB);
OCIBindByName($stmt, ’:key’, $key);
OCIBindByName($stmt, ’:data’, $clob, -1, OCI_B_CLOB);
$clob->WriteTemporary($data);
OCIExecute($stmt, OCI_DEFAULT);
OCICommit($conn);
$clob->close();
$clob->free();
OCIFreeStatement($stmt);
?>
```

OCINLogon (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Stabilisce, una nuova connessione a Oracle.

int **OCINLogon** (string username, string password [, string db]) \linebreak

OCINLogon() crea una nuova connessione a un database Oracle 8 e si autentica. Il terzo parametro opzionale può contenere il nome della istanza Oracle locale o il nome della voce in tnsnames.ora a cui ci si vuole connettere. Se il terzo parametro opzionale non è specificato, PHP usa la variabile d’ambiente ORACLE_SID (istanza di Oracle) o TWO_TASK (in tnsnames.ora) per determinare a quale database collegarsi.

OCINLogon() forza una nuova connessione. Si deve usare quando si ha necessità di isolare un insieme di transazioni. Di default, le connessioni sono condivise a livello di pagina se si usa **OCILogon()** o a livello di processo del web server se si usa **OCIPLogon()**. Se ci sono connessioni multiple aperte con **OCINLogon()**, tutti i commit e rollback avverranno solo sulla connessione specificata.

Questo esempio dimostra come le connessioni sono isolate.

Esempio 1. OCINLogon

```
<?php
print "<HTML><PRE>";
$db = "";

$c1 = ocilogon("scott","tiger",$db);
$c2 = ocinlogon("scott","tiger",$db);

function create_table($conn)
{ $stmt = ociparse($conn,"create table scott.hallo (test
varchar2(64))");
  ociexecute($stmt);
  echo $conn." created table\n\n";
}

function drop_table($conn)
{ $stmt = ociparse($conn,"drop table scott.hallo");
  ociexecute($stmt);
  echo $conn." dropped table\n\n";
}

function insert_data($conn)
{ $stmt = ociparse($conn,"insert into scott.hallo
            values('$conn' || ' ' || to_char(sysdate,'DD-MON-YY HH24:MI:SS'))");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." inserted hallo\n\n";
}

function delete_data($conn)
{ $stmt = ociparse($conn,"delete from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn." deleted hallo\n\n";
}

function commit($conn)
{ ocicommit($conn);
  echo $conn." committed\n\n";
}

function rollback($conn)
{ ocirollback($conn);
  echo $conn." rollback\n\n";
}

function select_data($conn)
{ $stmt = ociparse($conn,"select * from scott.hallo");
  ociexecute($stmt,OCI_DEFAULT);
  echo $conn."----selecting\n\n";
  while (ocifetch($stmt))
    echo $conn." <".ociresult($stmt,"TEST").">\n\n";
  echo $conn."----done\n\n";
}
```

```
create_table($c1);
insert_data($c1);

select_data($c1);
select_data($c2);

rollback($c1);

select_data($c1);
select_data($c2);

insert_data($c2);
commit($c2);

select_data($c1);

delete_data($c1);
select_data($c1);
select_data($c2);
commit($c1);

select_data($c1);
select_data($c2);

drop_table($c1);
print "</PRE></HTML>";
?>
```

Vedere anche **OCILogon()** e **OCIPLogon()**.

OCINumCols (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Restituisce il numero di campi che risultano da un comando SQL

int **OCINumCols** (int stmt) \linebreak

OCINumCols() restituisce il numero di campi contenuti in un’istruzione SQL.

Esempio 1. OCINumCols

```
<?php
    print "<HTML><PRE>\n";
    $conn = OCILogon("scott", "tiger");
    $stmt = OCIParse($conn,"select * from emp");
    OCIExecute($stmt);
    while ( OCIFetch($stmt) ) {
        print "\n";
        $ncols = OCINumCols($stmt);
        for ( $i = 1; $i <= $ncols; $i++ ) {
            $column_name  = OCIColumnName($stmt,$i);
```

```
            $column_value = OCIResult($stmt,$i);
            print $column_name . ': ' . $column_value . "\n";
        }
        print "\n";
    }
    OCIFreeStatement($stmt);
    OCILogoff($conn);
    print "</PRE>";
    print "</HTML>\n";
?>
```

OCIParse (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Analizza una query e restituisce un'istruzione.

int **OCIParse** (int conn, string query) \linebreak

OCIParse() analizza la *query* rispetto alla connessione *conn*. Restituisce un identificatore di istruzione se la query è valida, FALSE altrimenti. La *query* può essere qualsiasi comando SQL o blocco di codice PL/SQL.

OCIPLogon (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Stabilisce una connessione permanente a Oracle.

int **OCIPLogon** (string username, string password [, string db]) \linebreak

OCIPLogon() crea una connessione persistente a un database Oracle 8 e si autentica. Il terzo parametro opzionale può contenere il nome della istanza Oracle locale o il nome della voce in tnsnames.ora a cui ci si vuole connettere. Se il terzo parametro opzionale non ì specificato, PHP usa la variabile d'ambiente ORACLE_SID (istanza di Oracle) o TWO_TASK (in tnsnames.ora) per determinare a quale database collegarsi.

Vedere anche **OCILogon()** e **OCINLogon()**.

OCIResult (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Restituisce il valore di campo della tupla estratta

mixed **OCIResult** (int statement, mixed column) \linebreak

OCIResult() restituisce i dati del campo *column* nella tupla corrente (vedere **OCIFetch()**).**OCIResult()** restituirà tutto come stringa, eccetto i tipi astratti (ROWIDs, LOBs e FILEs).

OCIRollback (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Annulla le transazioni in sospeso

int **OCIRollback** (int connection) \linebreak

OCIRollback() annulla tutte le transazioni in sospeso sulla connessione Oracle `connection`.

OCIRowCount (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Restituisce il numero di tuple modificate

int **OCIRowCount** (int statement) \linebreak

OCIRowCounts() restituisce il numero di tuple modificate, ad esempio, da un comando update. Questa funzione non riporta il numero di tuple restituite da una select!

Esempio 1. OCIRowCount

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $stmt = OCIParse($conn,"create table emp2 as select * from emp");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows inserted.<BR>";
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"delete from emp2");
    OCIExecute($stmt);
    print OCIRowCount($stmt) . " rows deleted.<BR>";
    OCICommit($conn);
    OCIFreeStatement($stmt);
    $stmt = OCIParse($conn,"drop table emp2");
    OCIExecute($stmt);
    OCIFreeStatement($stmt);
    OCILogOff($conn);
    print "</PRE></HTML>";
?>
```

OCISaveLob (PHP 4 >= 4.0.0)

Coming soon

string **OCISaveLob** (object lob) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCISaveLobFile (PHP 4 >= 4.0.0)

Coming soon

string **OCISaveLobFile** (object lob) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

OCIServerVersion (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Restituisce una stringa contenente informazioni sulla versione del server

string **OCIServerVersion** (int conn) \linebreak

Esempio 1. OCIServerVersion

```
<?php
   $conn = OCILogon("scott","tiger");
   print "Server Version: " . OCIServerVersion($conn);
   OCILogOff($conn);
?>
```

OCISetPrefetch (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Imposta il numero di tuple da precaricare

int **OCISetPrefetch** (int stmt, int rows) \linebreak

Imposta il nomero di tuple da precaricare. Il valore di default è 1.

OCIStatementType (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Restituisce il tipo di un’istruzione OCI

string **OCIStatementType** (int stmt) \linebreak

OCIStatementType() restituisce uno dei seguenti valori:

1. "SELECT"
2. "UPDATE"
3. "DELETE"
4. "INSERT"
5. "CREATE"
6. "DROP"
7. "ALTER"
8. "BEGIN"
9. "DECLARE"
10. "UNKNOWN"

Esempio 1. Esempi di OCIStatementType()

```
<?php
    print "<HTML><PRE>";
    $conn = OCILogon("scott","tiger");
    $sql  = "delete from emp where deptno = 10";

    $stmt = OCIParse($conn,$sql);
    if ( OCIStatementType($stmt) == "DELETE" ) {
        die "You are not allowed to delete from this table<BR>";
    }

    OCILogoff($conn);
    print "</PRE></HTML>";
?>
```

OCIWriteLobToFile (PHP 4 >= 4.0.0)

Coming soon

void **OCIWriteLobToFile** (object lob [, string filename [, int start [, int lenght]]]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

LXX. OpenSSL functions

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Introduction

This module uses the functions of OpenSSL (http://www.openssl.org/) for generation and verification of signatures and for sealing (encrypting) and opening (decrypting) data. PHP-4.0.4pl1 requires OpenSSL >= 0.9.6, but PHP-4.0.5 and greater with also work with OpenSSL >= 0.9.5.

> **Nota:** Please keep in mind that this extension is still considered experimental!

OpenSSL offers many features that this module currently doesn't support. Some of these may be added in the future.

Key/Certificate parameters

Quite a few of the openssl functions require a key or a certificate parameter. PHP 4.0.5 and earlier have to use a key or certificate resource returned by one of the openssl_get_xxx functions. Later versions may use one of the following methods:

- Certificates
  1. An X.509 resource returned from openssl_x509_read
  2. A string having the format `file://path/to/cert.pem`; the named file must contain a PEM encoded certificate
  3. A string containing the content of a certificate, PEM encoded
- Public/Private Keys
  1. A key resource returned from openssl_get_publickey() or openssl_get_privatekey()
  2. For public keys only: an X.509 resource
  3. A string having the format `file://path/to/file.pem` - the named file must contain a PEM encoded certificate/private key (it may contain both)
  4. A string containing the content of a certificate/key, PEM encoded
  5. For private keys, you may also use the syntax *array($key, $passphrase)* where $key represents a key specified using the file:// or textual content notation above, and $passphrase

represents a string containing the passphrase for that private key

Certificate Verification

When calling a function that will verify a signature/certificate, the *cainfo* parameter is an array containing file and directory names the specify the locations of trusted CA files. If a directory is specified, then it must be a correctly formed hashed directory as the **openssl** command would use.

PKCS7 Flags/Constants

The S/MIME functions make use of flags which are specified using a bitfield which can include one or more of the following values:

Tabella 1. PKCS7 CONSTANTS

Constant	Description
PKCS7_TEXT	adds text/plain content type headers to encrypted/signed message. If decrypting or verifying, it strips those headers from the output - if the decrypted or verified message is not of MIME type text/plain then an error will occur.
PKCS7_BINARY	normally the input message is converted to "canonical" format which is effectively using CR and LF as end of line: as required by the S/MIME specification. When this options is present, no translation occurs. This is useful when handling binary data which may not be in MIME format.
PKCS7_NOINTERN	when verifying a message, certificates (if any) included in the message are normally searched for the signing certificate. With this option only the certificates specified in the *extracerts* parameter of openssl_pkcs7_verify() are used. The supplied certificates can still be used as untrusted CAs however.
PKCS7_NOVERIFY	do not verify the signers certificate of a signed message.
PKCS7_NOCHAIN	do not chain verification of signers certificates: that is don’t use the certificates in the signed message as untrusted CAs.

Constant	**Description**
PKCS7_NOCERTS	when signing a message the signer's certificate is normally included - with this option it is excluded. This will reduce the size of the signed message but the verifier must have a copy of the signers certificate available locally (passed using the `extracerts` to openssl_pkcs7_verify() for example.
PKCS7_NOATTR	normally when a message is signed, a set of attributes are included which include the signing time and the supported symmetric algorithms. With this option they are not included.
PKCS7_DETACHED	When signing a message, use cleartext signing with the MIME type multipart/signed. This is the default if the `flags` parameter to openssl_pkcs7_sign() if you do not specify any flags. If you turn this option off, the message will be signed using opaque signing, which is more resistant to translation by mail relays but cannot be read by mail agents that do not support S/MIME.
PKCS7_NOSIGS	Don't try and verify the signatures on a message

Nota: These constants were added in 4.0.6.

openssl_csr_export (PHP 4 >= 4.2.0)

Exports a CSR to file or a var

bool **openssl_csr_export** (resource csr, string out [, bool notext]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_csr_export_to_file (PHP 4 >= 4.2.0)

Exports a CSR to file or a var

bool **openssl_csr_export_to_file** (resource csr, string outfilename [, bool notext]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_csr_new (PHP 4 >= 4.2.0)

Generates a privkey and CSR

bool **openssl_csr_new** (array dn, resource privkey [, array extraattribs [, array configargs]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_csr_sign (PHP 4 >= 4.2.0)

Signs a cert with another CERT

resource **openssl_csr_sign** (mixed csr, mixed x509, mixed priv_key, long days) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_error_string (PHP 4 >= 4.0.6)

Return openSSL error message

mixed **openssl_error_string** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns an error message string, or FALSE if there are no more error messages to return.

openssl_error_string() returns the last error from the openSSL library. Error messages are stacked, so this function should be called multiple times to collect all of the information.

The parameters/return type of this function may change before it appears in a release version of PHP

Esempio 1. openssl_error_string() example

```
// lets assume you just called an openssl function that failed
while($msg = openssl_error_string())
    echo $msg . "<br />\n";
```

Nota: This function was added in 4.0.6.

openssl_free_key (PHP 4 >= 4.0.4)

Free key resource

void **openssl_free_key** (resource key_identifier) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

openssl_free_key() frees the key associated with the specified *key_identifier* from memory.

openssl_get_privatekey (PHP 4 >= 4.0.4)

Prepare a PEM formatted private key for use

resource **openssl_get_privatekey** (mixed key [, string passphrase]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns a positive key resource identifier on success, or `FALSE` on error.

openssl_get_privatekey() parses the PEM formatted private key specified by *`key`* and prepares it for use by other functions. The optional parameter *`passphrase`* must be used if the specified key is encrypted (protected by a passphrase).

openssl_get_publickey (PHP 4 >= 4.0.4)

Extract public key from certificate and prepare it for use

resource **openssl_get_publickey** (mixed certificate) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns a positive key resource identifier on success, or `FALSE` on error.

openssl_get_publickey() extracts the public key from an X.509 certificate specified by *`certificate`* and prepares it for use by other functions.

openssl_open (PHP 4 >= 4.0.4)

Open sealed data

bool **openssl_open** (string sealed_data, string open_data, string env_key, mixed priv_key_id) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento. If successful the opened data is returned in *`open_data`*.

openssl_open() opens (decrypts) *`sealed_data`* using the private key associated with the key identifier *`priv_key_id`* and the envelope key *`env_key`*, and fills *`open_data`* with the

decrypted data. The envelope key is generated when the data are sealed and can only be used by one specific private key. See openssl_seal() for more information.

Esempio 1. openssl_open() example

```
// $sealed and $env_key are assumed to contain the sealed data
// and our envelope key, both given to us by the sealer.

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);

// decrypt the data and store it in $open
if (openssl_open($sealed, $open, $env_key, $pkeyid))
    echo "here is the opened data: ", $open;
else
    echo "failed to open data";

// free the private key from memory
openssl_free_key($pkeyid);
```

See also openssl_seal().

openssl_pkcs7_decrypt (PHP 4 >= 4.0.6)

Decrypts an S/MIME encrypted message

bool **openssl_pkcs7_decrypt** (string infilename, string outfilename, mixed recipcert, mixed recipkey) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Decrypts the S/MIME encrypted message contained in the file specified by *`infilename`* using the certificate and it's associated private key specified by *`recipcert`* and *`recipkey`*.

The decrypted message is output to the file specified by *`outfilename`*

Esempio 1. openssl_pkcs7_decrypt() example

```
// $cert and $key are assumed to contain your personal certificate and private
// key pair, and that you are the recipient of an S/MIME message
$infilename = "encrypted.msg";  // this file holds your encrypted message
$outfilename = "decrypted.msg"; // make sure you can write to this file

if (openssl_pkcs7_decrypt($infilename, $outfilename, $cert, $key))
    echo "decrypted!";
else
    echo "failed to decrypt!";
```

Nota: This function was added in 4.0.6.

openssl_pkcs7_encrypt (PHP 4 >= 4.0.6)

Encrypt an S/MIME message

bool **openssl_pkcs7_encrypt** (string infile, string outfile, mixed recipcerts, array headers [, long flags]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

openssl_pkcs7_encrypt() takes the contents of the file named *infile* and encrypts them using an RC2 40-bit cipher so that they can only be read by the intended recipients specified by *recipcerts*, which is either a lone X.509 certificate, or an array of X.509 certificates. *headers* is an array of headers that will be prepended to the data after it has been encrypted. *flags* can be used to specify options that affect the encoding process - see PKCS7 constants. *headers* can be either an associative array keyed by header name, or an indexed array, where each element contains a single header line.

Esempio 1. openssl_pkcs7_encrypt() example

```
// the message you want to encrypt and send to your secret agent
// in the field, known as nighthawk.  You have his certificate
// in the file nighthawk.pem
$data = <<<EOD
Nighthawk,

Top secret, for your eyes only!
```

```
The enemy is closing in! Meet me at the cafe at 8.30am
to collect your forged passport!

HQ
EOD;

// load key
$key = implode("", file("nighthawk.pem"));

// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);

// encrypt it
if (openssl_pkcs7_encrypt("msg.txt", "enc.txt", $key,
    array("To" => "nighthawk@example.com", // keyed
syntax
          "From: HQ <hq@example.com>", // indexed syntax
          "Subject" => "Eyes only")))
{
    // message encrypted - send it!
    exec(ini_get("sendmail_path") . " < enc.txt");
}
```

openssl_pkcs7_sign (PHP 4 >= 4.0.6)

sign an S/MIME message

bool **openssl_pkcs7_sign** (string infilename, string outfilename, mixed signcert, mixed privkey, array headers [, long flags [, string extracertsfilename]]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

openssl_pkcs7_sign() takes the contents of the file named *infilename* and signs them using the certificate and it's matching private key specified by *signcert* and *privkey* parameters.

headers is an array of headers that will be prepended to the data after it has been signed (see openssl_pkcs7_encrypt() for more information about the format of this parameter.

flags can be used to alter the output - see PKCS7 constants - if not specified, it defaults to PKCS7_DETACHED.

extracerts specifies the name of a file containing a bunch of extra certificates to include in the signature which can for example be used to help the recipient to verify the certificate that you used.

Esempio 1. openssl_pkcs7_sign() example

```
// the message you want to sign so that recipient can be sure it was you that
// sent it
$data = <<<EOD

You have my authorization to spend $10,000 on dinner expenses.

The CEO
EOD;
// save message to file
$fp = fopen("msg.txt", "w");
fwrite($fp, $data);
fclose($fp);
// encrypt it
if (openssl_pkcs7_sign("msg.txt", "signed.txt", "mycert.pem",
    array("mycert.pem", "mypassphrase"),
    array("To" => "joes@sales.com", // keyed syntax
          "From: HQ <ceo@sales.com>", // indexed syntax
          "Subject" => "Eyes only"))
{
    // message signed - send it!
    exec(ini_get("sendmail_path") . " < signed.txt");
}
```

Nota: This function was added in 4.0.6.

openssl_pkcs7_verify (PHP 4 >= 4.0.6)

Verifies the signature of an S/MIME signed message

bool **openssl_pkcs7_verify** (string filename, int flags [, string outfilename [, array cainfo [, string extracerts]]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

openssl_pkcs7_verify() reads the S/MIME message contained in the filename specified by *filename* and examines the digital signature. It returns TRUE if the signature is verified, FALSE if it is not correct (the message has been tampered with, or the signing certificate is invalid), or -1 on error.

flags can be used to affect how the signature is verified - see PKCS7 constants for more information.

If the *outfilename* is specified, it should be a string holding the name of a file into which the certificates of the persons that signed the messages will be stored in PEM format.

If the *cainfo* is specified, it should hold information about the trusted CA certificates to use in the verification process - see certificate verification for more information about this parameter.

If the *extracerts* is specified, it is the filename of a file containing a bunch of certificates to use as untrusted CAs.

Nota: This function was added in 4.0.6.

openssl_pkey_export (PHP 4 >= 4.2.0)

Gets an exportable representation of a key into a string or file

bool **openssl_pkey_export** (mixed key, mixed out [, string passphrase [, array config_args]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_pkey_export_to_file (PHP 4 >= 4.2.0)

Gets an exportable representation of a key into a file

bool **openssl_pkey_export_to_file** (mixed key, string outfilename [, string passphrase [, array config_args]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_pkey_new (PHP 4 >= 4.2.0)

Generates a new private key

resource **openssl_pkey_new** ([array configargs]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_private_decrypt (PHP 4 >= 4.0.6)

Decrypts data with private key

bool **openssl_private_decrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_private_encrypt (PHP 4 >= 4.0.6)

Encrypts data with private key

bool **openssl_private_encrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_public_decrypt (PHP 4 >= 4.0.6)

Decrypts data with public key

bool **openssl_public_decrypt** (string data, string crypted, resource key [, int padding]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_public_encrypt (PHP 4 >= 4.0.6)

Encrypts data with public key

bool **openssl_public_encrypt** (string data, string crypted, mixed key [, int padding]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_seal (PHP 4 >= 4.0.4)

Seal (encrypt) data

int **openssl_seal** (string data, string sealed_data, array env_keys, array pub_key_ids) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns the length of the sealed data on success, or `FALSE` on error. If successful the sealed data is returned in *`sealed_data`*, and the envelope keys in *`env_keys`*.

openssl_seal() seals (encrypts) *`data`* by using RC4 with a randomly generated secret key. The key is encrypted with each of the public keys associated with the identifiers in *`pub_key_ids`* and each encrypted key is returned in *`env_keys`*. This means that one can send sealed data to multiple recipients (provided one has obtained their public keys). Each recipient must receive both the sealed data and the envelope key that was encrypted with the recipient's public key.

Esempio 1. openssl_seal() example

```
// $data is assumed to contain the data to be sealed

// fetch public keys for our recipients, and ready them
$fp = fopen("/src/openssl-0.9.6/demos/maurice/cert.pem", "r");
```

```
$cert = fread($fp, 8192);
fclose($fp);
$pk1 = openssl_get_publickey($cert);
// Repeat for second recipient
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pk2 = openssl_get_publickey($cert);

// seal message, only owners of $pk1 and $pk2 can decrypt $sealed with keys
// $ekeys[0] and $ekeys[1] respectively.
openssl_seal($data, $sealed, $ekeys, array($pk1,$pk2));

// free the keys from memory
openssl_free_key($pk1);
openssl_free_key($pk2);
```

See also openssl_open().

openssl_sign (PHP 4 >= 4.0.4)

Generate signature

bool **openssl_sign** (string data, string signature, mixed priv_key_id) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento. If successful the signature is returned in *signature*.

openssl_sign() computes a signature for the specified *data* by using SHA1 for hashing followed by encryption using the private key associated with *priv_key_id*. Note that the data itself is not encrypted.

Esempio 1. openssl_sign() example

```
// $data is assumed to contain the data to be signed

// fetch private key from file and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/key.pem", "r");
$priv_key = fread($fp, 8192);
fclose($fp);
$pkeyid = openssl_get_privatekey($priv_key);
```

```
// compute signature
openssl_sign($data, $signature, $pkeyid);

// free the key from memory
openssl_free_key($pkeyid);
```

See also openssl_verify().

openssl_verify (PHP 4 >= 4.0.4)

Verify signature

int **openssl_verify** (string data, string signature, mixed pub_key_id) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns 1 if the signature is correct, 0 if it is incorrect, and -1 on error.

openssl_verify() verifies that the *signature* is correct for the specified *data* using the public key associated with *pub_key_id*. This must be the public key corresponding to the private key used for signing.

Esempio 1. openssl_verify() example

```
// $data and $signature are assumed to contain the data and the signature

// fetch public key from certificate and ready it
$fp = fopen("/src/openssl-0.9.6/demos/sign/cert.pem", "r");
$cert = fread($fp, 8192);
fclose($fp);
$pubkeyid = openssl_get_publickey($cert);

// state whether signature is okay or not
$ok = openssl_verify($data, $signature, $pubkeyid);
if ($ok == 1)
    echo "good";
elseif ($ok == 0)
    echo "bad";
else
    echo "ugly, error checking signature";

// free the key from memory
openssl_free_key($pubkeyid);
```

See also openssl_sign().

openssl_x509_check_private_key (PHP 4 >= 4.2.0)

Checks if a private key corresponds to a CERT

bool **openssl_x509_check_private_key** (mixed cert, mixed key) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_x509_checkpurpose (PHP 4 >= 4.0.6)

Verifies if a certificate can be used for a particular purpose

bool **openssl_x509_checkpurpose** (mixed x509cert, int purpose, array cainfo [, string untrustedfile]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Returns `TRUE` if the certificate can be used for the intended purpose, `FALSE` if it cannot, or -1 on error.

openssl_x509_checkpurpose() examines the certificate specified by *x509cert* to see if it can be used for the purpose specified by *purpose*.

cainfo should be an array of trusted CA files/dirs as described in Certificate Verification.

untrustedfile, if specified, is the name of a PEM encoded file holding certificates that can be used to help verify the certificate, although no trust in placed in the certificates that come from that file.

Tabella 1. openssl_x509_checkpurpose() purposes

Constant	Description
X509_PURPOSE_SSL_CLIENT	Can the certificate be used for the client side of an SSL connection?
X509_PURPOSE_SSL_SERVER	Can the certificate be used for the server side of an SSL connection?
X509_PURPOSE_NS_SSL_SERVER	Can the cert be used for Netscape SSL server?
X509_PURPOSE_SMIME_SIGN	Can the cert be used to sign S/MIME email?
X509_PURPOSE_SMIME_ENCRYPT	Can the cert be used to encrypt S/MIME email?
X509_PURPOSE_CRL_SIGN	Can the cert be used to sign a certificate revocation list (CRL)?
X509_PURPOSE_ANY	Can the cert be used for Any/All purposes?

These options are not bitfields - you may specify one only!

Nota: This function was added in 4.0.6.

openssl_x509_export (PHP 4 >= 4.2.0)

Exports a CERT to file or a var

bool **openssl_x509_export** (mixed x509, string outfilename [, bool notext]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_x509_export_to_file (PHP 4 >= 4.2.0)

Exports a CERT to file or a var

bool **openssl_x509_export_to_file** (mixed x509, string outfilename [, bool notext]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

openssl_x509_free (PHP 4 >= 4.0.6)

Free certificate resource

void **openssl_x509_free** (resource x509cert) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

openssl_x509_free() frees the certificate associated with the specified *`x509cert`* resource from memory.

> **Nota:** This function was added in 4.0.6.

openssl_x509_parse (PHP 4 >= 4.0.6)

Parse an X509 certificate and return the information as an array

array **openssl_x509_parse** (mixed x509cert [, bool shortnames]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

openssl_x509_parse() returns information about the supplied *x509cert*, including fields such as subject name, issuer name, purposes, valid from and valid to dates etc. *shortnames* controls how the data is indexed in the array - if *shortnames* is TRUE (the default) then fields will be indexed with the short name form, otherwise, the long name form will be used - e.g.: CN is the shortname form of commonName.

The structure of the returned data is (deliberately) not yet documented, as it is still subject to change.

> **Nota:** This function was added in 4.0.6.

openssl_x509_read (PHP 4 >= 4.0.6)

Parse an X.509 certificate and return a resource identifier for it

resource **openssl_x509_read** (mixed x509certdata) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

openssl_x509_read() parses the certificate supplied by *x509certdata* and returns a resource identifier for it.

> **Nota:** This function was added in 4.0.6.

LXXI. Funzioni Oracle

Ora_Bind (PHP 3, PHP 4 >= 4.0.0)

effettua il binding di una variabile PHP ad un parametro di Oracle

int **ora_bind** (int cursor, string PHP variable name, string SQL parameter name, int length [, int type]) \linebreak

Restituisce `TRUE` se il binding riesce, altrimenti `FALSE`. I dettagli dell'errore si ottengono usando le funzioni ora_error() e ora_errorcode().

Questa funzione collega la variabile PHP indicata con un parametro SQL. Il parametro SQL deve essere nella forma ":name". Con il parametro facoltativo type, si può determinare se il parametro SQL è di in/out (0, default), in (1) oppure out (2). Dalla versione 3.0.1 di PHP, su possono utilizzare le costanti ORA_BIND_INOUT, ORA_BIND_IN e ORA_BIND_OUT al posto dei numeri.

ora_bind deve essere invocata dopo ora_parse() e prima di ora_exec(). I valori di input possono essere dati mediante assegnamento alle variabili PHP collegate; dopo aver chiamato ora_exec() le variabili PHP collegate contengono i valori di output, se disponibili.

```
<?php
ora_parse($curs, "declare tmp INTEGER; begin tmp := :in; :out := tmp; :x := 7.77; end;");
ora_bind($curs, "result", ":x", $len, 2);
ora_bind($curs, "input", ":in", 5, 1);
ora_bind($curs, "output", ":out", 5, 2);
$input = 765;
ora_exec($curs);
echo "Result: $result<BR>Out: $output<BR>In: $input";
?>
```

Ora_Close (PHP 3, PHP 4 >= 4.0.0)

chiude un cursore Oracle

int **ora_close** (int cursor) \linebreak

Restituisce `TRUE` se la chiusura è effettuata, altrimenti `FALSE`. I dettagli dell'errore si ottengono con le funzioni ora_error() e ora_errorcode().

Questa funzione chiude un corsore dati aperto con ora_open().

Ora_ColumnName (PHP 3, PHP 4 >= 4.0.0)

restituisce il nome di un campo risultato Oracle

string **Ora_ColumnName** (int cursor, int column) \linebreak

Restituisce il nome del campo/colonna *column* nel cursore *cursor*. Il nome è restituito in lettere maiuscole.

Ora_ColumnSize (PHP 3, PHP 4 >= 4.0.0)

restituisce la dimensione di un campo risultato Oracle

int **Ora_ColumnSize** (int cursor, int column) \linebreak

Restituisce la dimensione del campo/colonna *column* nel cursore *cursor*.

Ora_ColumnType (PHP 3, PHP 4 >= 4.0.0)

restituisce il tipo di un campo risultato Oracle

string **Ora_ColumnType** (int cursor, int column) \linebreak

Restituisce il tipo Oracle del campo/colonna *column* nel cursore *cursor*. Il tipo restituito sarà uno dei seguenti:

```
"VARCHAR2"
"VARCHAR"
"CHAR"
"NUMBER"
"LONG"
"LONG RAW"
"ROWID"
"DATE"
"CURSOR"
```

Ora_Commit (PHP 3, PHP 4 >= 4.0.0)

esegue una transazione Oracle

int **ora_commit** (int conn) \linebreak

Restituisce `TRUE` se l'operazione riesce, `FALSE` se si verifica un errore. I dettagli dell'errore si ottengono usando le funzioniora_error() e ora_errorcode().

Questa funzione esegue una transazione Oracle. Una transazione è definita come tutti i cambiamenti su una data connessione dall'ultimo commit/rollback, con autocommit spento, o dall'inizio della connessione.

Ora_CommitOff (PHP 3, PHP 4 >= 4.0.0)

disabilita il commit automatico

int **ora_commitoff** (int conn) \linebreak

Restituisce TRUE se l'operazione riesce, FALSE se si verifica un errore. I dettagli dell'errore si ottengono usando le funzioni ora_error() e ora_errorcode().

Questa funzione disabilita il commit automatico dopo ogni ora_exec().

Ora_CommitOn (PHP 3, PHP 4 >= 4.0.0)

abilita il commit automatico

int **ora_commiton** (int conn) \linebreak

Questa funzione abilita il commit automatico dopo ogni ora_exec() sulla connessione specificata.

Restituisce TRUE se l'operazione riesce, FALSE se si verifica un errore. I dettagli dell'errore si ottengono usando le funzioni ora_error() e ora_errorcode().

Ora_Do (PHP 3, PHP 4 >= 4.0.0)

Parse, Exec, Fetch

int **ora_do** (int conn, string query) \linebreak

Questa funzione è la combinazione veloce di ora_parse(), ora_exec() e ora_fetch(). Analizza, ed esegue un comando SQL, quindi scarica la prima tupla del risultato.

Restituisce TRUE se l'operazione riesce, FALSE se si verifica un errore. I dettagli dell'errore si ottengono usando le funzioni ora_error() e ora_errorcode().

Vedere anche ora_parse(),ora_exec(), e ora_fetch().

Ora_Error (PHP 3, PHP 4 >= 4.0.0)

restituisce il messaggio di errore di Oracle

string **Ora_Error** (int cursor_or_connection) \linebreak

Restituisce un messaggio d'errore nella forma *XXX-NNNNN* dove *XXX* è la sorgente dell'errore e *NNNNN* identifica il messaggio d'errore.

Nota: Il supporto per gli id di connessione è stato aggiunto nella versione 3.0.4.

Nelle versioni UNIX di Oracle, è possibile ottenere i dettagli dell'errore in questo modo: `$` **`oerr ora`** ***`00001`*** `00001, 00000, "unique constraint (%s.%s) violated" // *Cause: An update or insert statement attempted to insert a duplicate key // For Trusted ORACLE configured in DBMS MAC mode, you may see // this message if a duplicate entry exists at a different level. // *Action: Either remove the unique restriction or do not insert the key`

Ora_ErrorCode (PHP 3, PHP 4 >= 4.0.0)

restituisce il codice di errore di Oracle

int **Ora_ErrorCode** (int cursor_or_connection) \linebreak

Restituisce il codice numerico di errore dell'ultimo comando eseguito sul cursore o sulla connessione specificata.

> **Nota:** Il supporto per gli id di connessione è stato aggiunto nella versione 3.0.4.

Ora_Exec (PHP 3, PHP 4 >= 4.0.0)

esegue dei comandi già analizzati su un cursore Oracle

int **ora_exec** (int cursor) \linebreak

Restituisce `TRUE` se l'operazione riesce, `FALSE` se si verifica un errore. I dettagli dell'errore si ottengono usando le funzioni ora_error() e ora_errorcode().

Vedere anche ora_parse(), ora_fetch(), e ora_do().

Ora_Fetch (PHP 3, PHP 4 >= 4.0.0)

scarica una tupla di dati da un cursore

int **ora_fetch** (int cursor) \linebreak

Restituisce `TRUE` (la tupla è stata acquisita) o `FALSE` (non ci sono più tuple, o è avvenuto un errore). Se è avvenuto un errore, i dettagli si ottengono usando le funzioni ora_error() e ora_errorcode(). Se non sono avvenuti errori, ora_errorcode() restituirà 0.

Acquisisce una tupla di dati dal cursore specificato.

Vedere anche ora_parse(),ora_exec(), e ora_do().

Ora_Fetch_Into (PHP 3, PHP 4 >= 4.0.0)

Scarica una tupla nell'array specificato

int **ora_fetch_into** (int cursor, array result [, int flags]) \linebreak

Recupera una tupla e la mette in un array. Il parametro *`flags`* ha due valori: se il flag `ORA_FETCHINFO_NULLS` è impostato, i campi con valori `NULL` vengono inseriti nell'array; se il flag `ORA_FETCHINFO_ASSOC` è impostato, viene creato un array associativo.

Restituisce il numero di campi acquisiti.

Esempio 1. ora_fetch_into()()

```
<?php
$risultato = array();
ora_fetch_into($cursore, $results);
echo $risultato[0];
echo $risultato[1];
$risultato = array();
ora_fetch_into($cursore, $risultato, ORA_FETCHINTO_NULLS|ORA_FETCHINTO_ASSOC);
echo $risultato['MioCampo'];
?>
```

Vedere anche ora_parse(),ora_exec(), ora_fetch(), e ora_do().

Ora_GetColumn (PHP 3, PHP 4 >= 4.0.0)

restituisce i dati di un campo acquisito

mixed **ora_getcolumn** (int cursor, mixed column) \linebreak

Restituisce il contenuto del campo/colonna. Se avviene un errore, viene restituito il valore `FALSE` e ora_errorcode() restituirà un valore diverso da zero. Si noti, comunque, che un test al valore `FALSE` sul risultato di questa funzione può dare esito positivo anche in caso non ci siano errori (campo `NULL`, stringa vuota, il numero 0, la stringa "0").

Restituisce i dati del campo o del risultato di una funzione.

Ora_Logoff (PHP 3, PHP 4 >= 4.0.0)

chiude una connessione Oracle

int **ora_logoff** (int connection) \linebreak

Restituisce `TRUE` se l'operazione riesce, `FALSE` se si verifica un errore. I dettagli dell'errore si ottengono usando le funzioni ora_error() e ora_errorcode().

Depenna l'utente e disconnette dal server.

Vedere anche ora_logon().

Ora_Logon (PHP 3, PHP 4 >= 4.0.0)

apre una connessione Oracle

int **ora_logon** (string user, string password) \linebreak

Stabilisce una connessione tra PHP e un database Oracle utilizzando le username e password specificate.

Le connessioni possono essere create usando SQL*Net fornendo il nomeTNS a `user` in questo modo:

```
$conn = Ora_Logon("user<emphasis>@TNSNAME</emphasis>", "pass");
```

Se i dati contengono caratteri non-ASCII, si deve controllare di avere impostato NLS_LANG nel proprio environment. Per quanto riguarda i moduli del server, occorre impostarlo nell' environment del server prima di avviarlo.

Restituisce un indice di connessione se l'operazione ha success, oppure `FALSE` in caso di errore. I dettagli dell'erroresi ottengono usando le funzioni ora_error() e ora_errorcode() functions.

Ora_Numcols (PHP 3, PHP 4 >= 4.0.0)

Restituisce il numero di campi/colonne

int **ora_numcols** (int cursor_ind) \linebreak

ora_numcols() restituisce il numero di campi di un risultato. Dopo una sequenza parse/exec/fetch restituisce solo risultati che abbiano coerenza.

Vedere anche ora_parse(),ora_exec(), ora_fetch(), e ora_do().

Ora_Numrows (PHP 3, PHP 4 >= 4.0.0)

Restituisce il numero di tuple

int **ora_numrows** (int cursor_ind) \linebreak

ora_numrows() restituisce il numero di tuple in un risultato.

Ora_Open (PHP 3, PHP 4 >= 4.0.0)

apre un cursore Oracle

int **ora_open** (int connection) \linebreak

Apre un cursore Oracle associato alla connessione.

Restituisce un indice di cursore oppure `FALSE` in caso di errore. I dettagli dell'errore si ottengono usando le funzioni ora_error() e ora_errorcode().

Ora_Parse (PHP 3, PHP 4 >= 4.0.0)

analizza un comando SQL

int **ora_parse** (int cursor_ind, string sql_statement, int defer) \linebreak

Questa funziona analizza un comando SQL o un blocco PL/SQL e lo associa al cursore specificato.

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

Vedere anche ora_exec(), ora_fetch(), e ora_do().

Ora_pLogon (PHP 3, PHP 4 >= 4.0.0)

Apre una connessione Oracle permanente

int **ora_plogon** (string user, string password) \linebreak

Stabilisce una connessione permanente tra PHP ed un database Oracle con le username and password specificate.

Vedere anche ora_logon().

Ora_Rollback (PHP 3, PHP 4 >= 4.0.0)

esegue il rollback della transazione

int **ora_rollback** (int connection) \linebreak

Questa funzione annulla una transazione Oracle. (Vedere ora_commit() per la definizione di una transazione.)

Restituisce `TRUE` se l'operazione riesce, `FALSE` in caso di errore. I dettagli dell'errore si ottengono usando le funzioni ora_error() e ora_errorcode() functions.

LXXII. Ovrimos SQL functions

Ovrimos SQL Server, is a client/server, transactional RDBMS combined with Web capabilities and fast transactions.

Ovrimos SQL Server is available at www.ovrimos.com (http://www.ovrimos.com/). To enable ovrimos support in PHP just compile php with the '--with-ovrimos' parameter to configure script. You'll need to install the sqlcli library available in the Ovrimos SQL Server distribution.

Esempio 1. Connect to Ovrimos SQL Server and select from a system table

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo ("Connection ok!");
    $res = ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will just connect to SQL Server.

ovrimos_close (PHP 4 >= 4.0.3)

Closes the connection to ovrimos

void **ovrimos_close** (int connection) \linebreak

ovrimos_close() is used to close the specified connection.

ovrimos_close() closes a connection to Ovrimos. This has the effect of rolling back uncommitted transactions.

ovrimos_commit (PHP 4 >= 4.0.3)

Commits the transaction

int **ovrimos_commit** (int connection_id) \linebreak

ovrimos_commit() is used to commit the transaction.

ovrimos_commit() commits the transaction.

ovrimos_connect (PHP 4 >= 4.0.3)

Connect to the specified database

int **ovrimos_connect** (string host, string db, string user, string password) \linebreak

ovrimos_connect() is used to connect to the specified database.

ovrimos_connect() returns a connection id (greater than 0) or 0 for failure. The meaning of 'host' and 'port' are those used everywhere in Ovrimos APIs. 'Host' is a host name or IP address and 'db' is either a database name, or a string containing the port number.

Esempio 1. ovrimos_connect() Example

```
<?php
$conn = ovrimos_connect ("server.domain.com", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

The above example will connect to the database and print out the specified table.

ovrimos_cursor (PHP 4 >= 4.0.3)

Returns the name of the cursor

int **ovrimos_cursor** (int result_id) \linebreak

ovrimos_cursor() is used to get the name of the cursor.

ovrimos_cursor() returns the name of the cursor. Useful when wishing to perform positioned updates or deletes.

ovrimos_exec (PHP 4 >= 4.0.3)

Executes an SQL statement

int **ovrimos_exec** (int connection_id, string query) \linebreak

ovrimos_exec() is used to execute an SQL statement.

ovrimos_exec() executes an SQL statement (query or update) and returns a result_id or `FALSE`. Evidently, the SQL statement should not contain parameters.

ovrimos_execute (PHP 4 >= 4.0.3)

Executes a prepared SQL statement

bool **ovrimos_execute** (int result_id [, array parameters_array]) \linebreak

ovrimos_execute() is used to execute an SQL statement.

ovrimos_execute() executes a prepared statement. Returns `TRUE` or `FALSE`. If the prepared statement contained parameters (question marks in the statement), the correct number of parameters should be passed in an array. Notice that I don't follow the PHP convention of placing just the name of the optional parameter inside square brackets. I couldn't bring myself on liking it.

ovrimos_fetch_into (PHP 4 >= 4.0.3)

Fetches a row from the result set

bool **ovrimos_fetch_into** (int result_id, array result_array [, string how [, int rownumber]]) \linebreak

ovrimos_fetch_into() is used to fetch a row from the result set.

ovrimos_fetch_into() fetches a row from the result set into 'result_array', which should be passed by reference. Which row is fetched is determined by the two last parameters. 'how' is one of 'Next' (default), 'Prev', 'First', 'Last', 'Absolute', corresponding to forward direction from current

position, backward direction from current position, forward direction from the start, backward direction from the end and absolute position from the start (essentially equivalent to 'first' but needs 'rownumber'). Case is not significant. 'Rownumber' is optional except for absolute positioning. Returns TRUE or FALSE.

Esempio 1. A fetch into example

```
<?php
$conn=ovrimos_connect ("neptune", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn,"select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_into ($res, &$row)) {
            list ($table_id, $table_name) = $row;
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_into ($res, &$row)) {
                list ($table_id, $table_name) = $row;
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This example will fetch a row.

ovrimos_fetch_row (PHP 4 >= 4.0.3)

Fetches a row from the result set

bool **ovrimos_fetch_row** (int result_id [, int how [, int row_number]]) \linebreak

ovrimos_fetch_row() is used to fetch a row from the result set.

ovrimos_fetch_row() fetches a row from the result set. Column values should be retrieved with other calls. Returns TRUE or FALSE.

Esempio 1. A fetch row example

```
<?php
$conn = ovrimos_connect ("remote.host", "8001", "admin", "password");
```

```
if ($conn != 0) {
    echo "Connection ok!";
    $res=ovrimos_exec ($conn, "select table_id, table_name from sys.tables");
    if ($res != 0) {
        echo "Statement ok!";
        if (ovrimos_fetch_row ($res, "First")) {
            $table_id = ovrimos_result ($res, 1);
            $table_name = ovrimos_result ($res, 2);
            echo "table_id=".$table_id.", table_name=".$table_name."\n";
            if (ovrimos_fetch_row ($res, "Next")) {
                $table_id = ovrimos_result ($res, "table_id");
                $table_name = ovrimos_result ($res, "table_name");
                echo "table_id=".$table_id.", table_name=".$table_name."\n";
            } else {
                echo "Next: error\n";
            }
        } else {
            echo "First: error\n";
        }
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

This will fetch a row and print the result.

ovrimos_field_len (PHP 4 >= 4.0.3)

Returns the length of the output column

int **ovrimos_field_len** (int result_id, int field_number) \linebreak

ovrimos_field_len() is used to get the length of the output column with number *field_number*, in result *result_id*.

ovrimos_field_len() returns the length of the output column at the (1-based) index specified.

ovrimos_field_name (PHP 4 >= 4.0.3)

Returns the output column name

int **ovrimos_field_name** (int result_id, int field_number) \linebreak

ovrimos_field_name() is used to get the output column name.

ovrimos_field_name() returns the output column name at the (1-based) index specified.

ovrimos_field_num (PHP 4 >= 4.0.3)

Returns the (1-based) index of the output column

int **ovrimos_field_num** (int result_id, string field_name) \linebreak

ovrimos_field_num() is used to get the (1-based) index of the output column.

ovrimos_field_num() returns the (1-based) index of the output column specified by name, or `FALSE`.

ovrimos_field_type (PHP 4 >= 4.0.3)

Returns the (numeric) type of the output column

int **ovrimos_field_type** (int result_id, int field_number) \linebreak

ovrimos_field_type() is used to get the (numeric) type of the output column.

ovrimos_field_type() returns the (numeric) type of the output column at the (1-based) index specified.

ovrimos_free_result (PHP 4 >= 4.0.3)

Frees the specified result_id

bool **ovrimos_free_result** (int result_id) \linebreak

ovrimos_free_result() is used to free the result_id.

ovrimos_free_result() frees the specified result_id *`result_id`*. Returns `TRUE`.

ovrimos_longreadlen (PHP 4 >= 4.0.3)

Specifies how many bytes are to be retrieved from long datatypes

int **ovrimos_longreadlen** (int result_id, int length) \linebreak

ovrimos_longreadlen() is used to specify how many bytes are to be retrieved from long datatypes.

ovrimos_longreadlen() specifies how many bytes are to be retrieved from long datatypes (long varchar and long varbinary). Default is zero. It currently sets this parameter the specified result set. Returns `TRUE`.

ovrimos_num_fields (PHP 4 >= 4.0.3)

Returns the number of columns

int **ovrimos_num_fields** (int result_id) \linebreak

ovrimos_num_fields() is used to get the number of columns.

ovrimos_num_fields() returns the number of columns in a result_id resulting from a query.

ovrimos_num_rows (PHP 4 >= 4.0.3)

Returns the number of rows affected by update operations

int **ovrimos_num_rows** (int result_id) \linebreak

ovrimos_num_rows() is used to get the number of rows affected by update operations.

ovrimos_num_rows() returns the number of rows affected by update operations.

ovrimos_prepare (PHP 4 >= 4.0.3)

Prepares an SQL statement

int **ovrimos_prepare** (int connection_id, string query) \linebreak

ovrimos_prepare() is used to prepare an SQL statement.

ovrimos_prepare() prepares an SQL statement and returns a result_id (or FALSE on failure).

Esempio 1. Connect to Ovrimos SQL Server and prepare a statement

```
<?php
$conn=ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn!=0) {
    echo "Connection ok!";
    $res=ovrimos_prepare ($conn, "select table_id, table_name
                              from sys.tables where table_id=1");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res)) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close($conn);
}
?>
```

This will connect to Ovrimos SQL Server, prepare a statement and the execute it.

ovrimos_result (PHP 4 >= 4.0.3)

Retrieves the output column

int **ovrimos_result** (int result_id, mixed field) \linebreak

ovrimos_result() is used to retrieve the output column.

ovrimos_result() retrieves the output column specified by 'field', either as a string or as an 1-based index.

ovrimos_result_all (PHP 4 >= 4.0.3)

Prints the whole result set as an HTML table

bool **ovrimos_result_all** (int result_id [, string format]) \linebreak

ovrimos_result_all() is used to print an HTML table containing the whole result set.

ovrimos_result_all() prints the whole result set as an HTML table. Returns `TRUE` or `FALSE`.

Esempio 1. Prepare a statement, execute, and view the result

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_prepare ($conn, "select table_id, table_name
                                    from sys.tables where table_id = 7");
    if ($res != 0) {
        echo "Prepare ok!";
        if (ovrimos_execute ($res, array(3))) {
            echo "Execute ok!\n";
            ovrimos_result_all ($res);
        } else {
            echo "Execute not ok!";
        }
        ovrimos_free_result ($res);
    } else {
        echo "Prepare not ok!\n";
    }
    ovrimos_close($conn);
}
?>
```

This will execute an SQL statement and print the result in an HTML table.

Esempio 2. Ovrimos_result_all with meta-information

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "select table_id, table_name
                                 from sys.tables where table_id = 1")
    if ($res != 0) {
        echo "Statement ok! cursor=".ovrimos_cursor ($res)."\n";
        $colnb = ovrimos_num_fields ($res);
        echo "Output columns=".$colnb."\n";
        for ($i=1; $i <= $colnb; $i++) {
            $name = ovrimos_field_name ($res, $i);
            $type = ovrimos_field_type ($res, $i);
            $len = ovrimos_field_len ($res, $i);
            echo "Column ".$i." name=".$name." type=".$type." len=".$len."\n";
        }
        ovrimos_result_all ($res);
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

Esempio 3. ovrimos_result_all example

```
<?php
$conn = ovrimos_connect ("db_host", "8001", "admin", "password");
if ($conn != 0) {
    echo "Connection ok!";
    $res = ovrimos_exec ($conn, "update test set i=5");
    if ($res != 0) {
        echo "Statement ok!";
        echo ovrimos_num_rows ($res)." rows affected\n";
        ovrimos_free_result ($res);
    }
    ovrimos_close($conn);
}
?>
```

ovrimos_rollback (PHP 4 >= 4.0.3)

Rolls back the transaction

int **ovrimos_rollback** (int connection_id) \linebreak

ovrimos_rollback() is used to roll back the transaction.

ovrimos_rollback() rolls back the transaction.

LXXIII. Output Control Functions

The Output Control functions allow you to control when output is sent from the script. This can be useful in several different situations, especially if you need to send headers to the browser after your script has began outputting data. The Output Control functions do not affect headers sent using header() or setcookie(), only functions such as echo() and data between blocks of PHP code.

Esempio 1. Output Control example

```
<?php

ob_start();
echo "Hello\n";

setcookie ("cookiename", "cookiedata");

ob_end_flush();

?>
```

In the above example, the output from echo() would be stored in the output buffer until ob_end_flush() was called. In the mean time, the call to setcookie() successfully stored a cookie without causing an error. (You can not normally send headers to the browser after data has already been sent.)

See also header() and setcookie().

flush (PHP 3, PHP 4 >= 4.0.0)

Flush the output buffer

void **flush** (void) \linebreak

Flushes the output buffers of PHP and whatever backend PHP is using (CGI, a web server, etc). This effectively tries to push all the output so far to the user's browser.

> **Nota: flush()** has no effect on the buffering scheme of your webserver or the browser on the client side.
>
> Several servers, especially on Win32, will still buffer the output from your script until it terminates before transmitting the results to the browser.
>
> Server modules for Apache like mod_gzip may do buffering of their own that will cause **flush()** to not result in data being sent immediately to the client.
>
> Even the browser may buffer its input before displaying it. Netscape, for example, buffers text until it receives an end-of-line or the beginning of a tag, and it won't render tables until the </table> tag of the outermost table is seen.
>
> Some versions of Microsoft Internet Explorer will only start to display the page after they have received 256 bytes of output, so you may need to send extra whitespace before flushing to get those browsers to display the page.

ob_clean (PHP 4 >= 4.2.0)

Clean (erase) the output buffer

void **ob_clean** (void) \linebreak

This function discards the contents of the output buffer.

This function does not destroy the output buffer like ob_end_clean() does.

See also ob_flush(), ob_end_flush() and ob_end_clean().

ob_end_clean (PHP 4 >= 4.0.0)

Clean (erase) the output buffer and turn off output buffering

void **ob_end_clean** (void) \linebreak

This function discards the contents of the output buffer and turns off output buffering.

See also ob_start(), ob_clean() and ob_end_flush().

ob_end_flush (PHP 4 >= 4.0.0)

Flush (send) the output buffer and turn off output buffering

void **ob_end_flush** (void) \linebreak

This function will send the contents of the output buffer (if any) and turn output buffering off. If you want to further process the buffer's contents you have to call ob_get_contents() before **ob_end_flush()** as the buffer contents are discarded after **ob_end_flush()** is called.

See also ob_start(), ob_get_contents(), ob_flush() and ob_end_clean().

ob_flush (PHP 4 >= 4.2.0)

Flush (send) the output buffer

void **ob_flush** (void) \linebreak

This function will send the contents of the output buffer (if any). If you want to further process the buffer's contents you have to call ob_get_contents() before **ob_flush()** as the buffer contents are discarded after **ob_flush()** is called.

This function does not destroy the output buffer like ob_end_flush() does.

See also ob_get_contents(), ob_clean(), ob_end_flush() and ob_end_clean().

ob_get_contents (PHP 4 >= 4.0.0)

Return the contents of the output buffer

string **ob_get_contents** (void) \linebreak

This will return the contents of the output buffer or `FALSE`, if output buffering isn't active.

See also ob_start() and ob_get_length().

ob_get_length (PHP 4 >= 4.0.2)

Return the length of the output buffer

string **ob_get_length** (void) \linebreak

This will return the length of the contents in the output buffer or `FALSE`, if output buffering isn't active.

See also ob_start() and ob_get_contents().

ob_get_level (PHP 4 >= 4.2.0)

Return the nesting level of the output buffering mechanism

int **ob_get_level** (void) \linebreak

This will return the level of nested output buffering handlers.

See also ob_start() and ob_get_contents().

ob_gzhandler (PHP 4 >= 4.0.4)

ob_start callback function to gzip output buffer

string **ob_gzhandler** (string buffer [, int mode]) \linebreak

> **Nota:** *mode* was added in PHP 4.0.5.

ob_gzhandler() is intended to be used as a callback function for ob_start() to help facilitate sending gz-encoded data to web browsers that support compressed web pages. Before **ob_gzhandler()** actually sends compressed data, it determines what type of content encoding the browser will accept ("gzip", "deflate" or none at all) and will return it’s output accordingly. All browsers are supported since it’s up to the browser to send the correct header saying that it accepts compressed web pages.

Esempio 1. ob_gzhandler() Example

```
<?php

ob_start("ob_gzhandler");

?>
<html>
<body>
<p>This should be a compressed page.
</html>
<body>
```

See also ob_start() and ob_end_flush().

ob_implicit_flush (PHP 4 >= 4.0.0)

Turn implicit flush on/off

void **ob_implicit_flush** ([int flag]) \linebreak

ob_implicit_flush() will turn implicit flushing on or off (if no *flag* is given, it defaults to on). Implicit flushing will result in a flush operation after every output call, so that explicit calls to flush() will no longer be needed.

Turning implicit flushing on will disable output buffering, the output buffers current output will be sent as if ob_end_flush() had been called.

See also flush(), ob_start(), and ob_end_flush().

ob_start (PHP 4 >= 4.0.0)

Turn on output buffering

void **ob_start** ([string output_callback]) \linebreak

This function will turn output buffering on. While output buffering is active no output is sent from the script (other than headers), instead the output is stored in an internal buffer.

The contents of this internal buffer may be copied into a string variable using ob_get_contents(). To output what is stored in the internal buffer, use ob_end_flush(). Alternatively, ob_end_clean() will silently discard the buffer contents.

An optional *output_callback* function may be specified. This function takes a string as a parameter and should return a string. The function will be called when ob_end_flush() is called, or when the output buffer is flushed to the browser at the end of the request. When *output_callback* is called, it will receive the contents of the output buffer as its parameter and is expected to return a new output buffer as a result, which will be sent to the browser.

> **Nota:** In PHP 4.0.4, ob_gzhandler() was introduced to facilitate sending gz-encoded data to web browsers that support compressed web pages. ob_gzhandler() determines what type of content encoding the browser will accept and will return it's output accordingly.

Output buffers are stackable, that is, you may call **ob_start()** while another **ob_start()** is active. Just make sure that you call ob_end_flush() the appropriate number of times. If multiple output callback functions are active, output is being filtered sequentially through each of them in nesting order.

ob_end_clean(), ob_end_flush(), ob_clean(), ob_flush() and **ob_start()** may not be called from a callback function. If you call them from callback function, the behavior is undefined. If you would like to delete the contents of a buffer, return "" (a null string) from callback function.

Esempio 1. User defined callback function example

```
<?php

function callback($buffer) {

  // replace all the apples with oranges
  return (ereg_replace("apples", "oranges", $buffer));

}

ob_start("callback");
```

```
?>

<html>
<body>
<p>It’s like comparing apples to oranges.
</body>
</html>

<?php

ob_end_flush();

?>
```

Would produce:

```
<html>
<body>
<p>It’s like comparing oranges to oranges.
</body>
</html>
```

See also ob_get_contents(), ob_end_flush(), ob_end_clean(), ob_implicit_flush() and ob_gzhandler().

LXXIV. Proprietà object e method call overloading

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Lo scopo di questa estensione è di permettere l'overloading delle proprietà di accesso agli oggetti e dei metodi di chiamata. Solo una funzione è definita in questa estensione, overload() che prende il nome dalla classe che ha questa funzionalità abilitata. La classe nominata ha da definire metodi appropriati se vuole avere questa funzionalità: `__get()`, `__set()` and `__call()` rispettivamente per ricevere/impostare una proprietà, o chiamare un metodo. Questa strada del sovraccarico può essere selettiva. Dentro queste funzioni handler l'overloading è disabilitato così si può accedere alle proprietà dell'oggetto normalmente.

Alcuni semplici esempi sull'uso della funzione overload()

Esempio 1. Overloading di una classe PHP

```
<?php

class OO
{
    var $a = 111;
    var $elem = array('b' => 9, 'c' => 42);

    // Callback method for getting a property
    function __get($prop_name, &$prop_value)
    {
        if (isset($this->elem[$prop_name])) {
            $prop_value = $this->elem[$prop_name];
            return true;
        } else {
            return false;
        }
    }

    // Callback method for setting a property
    function __set($prop_name, $prop_value)
    {
        $this->elem[$prop_name] = $prop_value;
        return true;
    }
}

// Here we overload the OO object
overload('OO');

$o = new OO;
print "\$o->a: $o->a\n"; // print: $o->a:
```

```
print "\$o->b: $o->b\n"; // print: $o->b: 9
print "\$o->c: $o->c\n"; // print: $o->c: 42
print "\$o->d: $o->d\n"; // print: $o->d:

// add a new item to the $elem array in OO
$o->x = 56;

// instantiate stdclass (it is built-in in PHP 4)
// $val is not overloaded!
$val = new stdclass;
$val->prop = 555;

// Set "a" to be an array with the $val object in it
// But __set() will put this in the $elem array
$o->a = array($val);
var_dump($o->a[0]->prop);

?>
```

Attenzione

Siccome è una estensione sperimentale, non tutto funziona. Non c'è attualmente il supporto per `__call()`, puoi solo overloadare le operazioni di ricezione e di impostazione per le proprietà. Non puoi invocare l'overloading handlers originale della classe, e `__set()` funziona solo su un livello di proprietà di accesso.

overload (PHP 4 >= 4.2.0)

Abilita le proprietà e il method call overloading per una classe

void **overload** ([string class_name]) \linebreak

La funzione **overload()** abiliterà la proprietà e il method call overloading per una classe identificata dal parametro `class_name`. Guarda un esempio nella sezione di introduzione di questa parte..

LXXV. PDF functions

Introduction

The PDF functions in PHP can create PDF files using the PDFlib library created by Thomas Merz (http://www.pdflib.com/corporate/tm.html). PDFlib is available for download at http://www.pdflib.com/pdflib/index.html, but requires that you purchase a license for commercial use. The JPEG (ftp://ftp.uu.net/graphics/jpeg/) and TIFF (http://www.libtiff.org/) libraries are required to compile this extension. Please see the PDFlib installation section for more information about compiling PDF support into PHP.

The documentation in this section is only meant to be an overview of the available functions in the PDFlib library and should not be considered an exhaustive reference. Please consult the documentation included in the source distribution of PDFlib for the full and detailed explanation of each function here. It provides a very good overview of what PDFlib is capable of doing and contains the most up-to-date documentation of all functions.

All of the functions in PDFlib and the PHP module have identical function names and parameters. You will need to understand some of the basic concepts of PDF and PostScript to efficiently use this extension. All lengths and coordinates are measured in PostScript points. There are generally 72 PostScript points to an inch, but this depends on the output resolution. Please see the PDFlib documentation included with the source distribution of PDFlib for a more thorough explanation of the coordinate system used.

Please note that most of the PDF functions require a *pdf object* as it's first parameter. Please see the examples below for more information.

> **Nota:** An alternative PHP module for PDF document creation based on FastIO's (http://www.fastio.com/) ClibPDF is available. Please see the ClibPDF section for details. Note that ClibPDF has a slightly different API compared to PDFlib.

Confusion with old PDFlib versions

Starting with PHP 4.0.5, the PHP extension for PDFlib is officially supported by PDFlib GmbH. This means that all the functions described in the PDFlib manual (V3.00 or greater) are supported by PHP 4 with exactly the same meaning and the same parameters. Only the return values may differ from the PDFlib manual, because the PHP convention of returning FALSE was adopted. For compatibility reasons this binding for PDFlib still supports the old functions, but they should be replaced by their new versions. PDFlib GmbH will not support any problems arising from the use of these deprecated functions.

Tabella 1. Deprecated functions and its replacements

Old function	Replacement
pdf_put_image()	Not needed anymore.
pdf_execute_image()	Not needed anymore.
pdf_get_annotation()	**pdf_get_bookmark()** using the same parameters.

Old function	Replacement
pdf_get_font()	pdf_get_value() passing `"font"` as the second parameter.
pdf_get_fontsize()	pdf_get_value() passing `"fontsize"` as the second parameter.
pdf_get_fontname()	pdf_get_parameter() passing `"fontname"` as the second parameter.
pdf_set_info_creator()	pdf_set_info() passing `"Creator"` as the second parameter.
pdf_set_info_title()	pdf_set_info() passing `"Title"` as the second parameter.
pdf_set_info_subject()	pdf_set_info() passing `"Subject"` as the second parameter.
pdf_set_info_author()	pdf_set_info() passing `"Author"` as the second parameter.
pdf_set_info_keywords()	pdf_set_info() passing `"Keywords"` as the second parameter.
pdf_set_leading()	pdf_set_value() passing `"leading"` as the second parameter.
pdf_set_text_rendering()	pdf_set_value() passing `"textrendering"` as the second parameter.
pdf_set_text_rise()	pdf_set_value() passing `"textrise"` as the second parameter.
pdf_set_horiz_scaling()	pdf_set_value() passing `"horizscaling"` as the second parameter.
pdf_set_text_matrix()	Not available anymore
pdf_set_char_spacing()	pdf_set_value() passing `"charspacing"` as the second parameter.
pdf_set_word_spacing()	pdf_set_value() passing `"wordspacing"` as the second parameter.
pdf_set_transition()	pdf_set_parameter() passing `"transition"` as the second parameter.
pdf_open()	pdf_new() plus an subsequent call of pdf_open_file()
pdf_set_font()	pdf_findfont() plus an subsequent call of pdf_setfont()
pdf_set_duration()	pdf_set_value() passing `"duration"` as the second parameter.
pdf_open_gif()	pdf_open_image_file() passing `"gif"` as the second parameter.
pdf_open_jpeg()	pdf_open_image_file() passing `"jpeg"` as the second parameter.
pdf_open_tiff()	pdf_open_image_file() passing `"tiff"` as the second parameter.
pdf_open_png()	pdf_open_image_file() passing `"png"` as the second parameter.

Old function	Replacement
pdf_get_image_width()	pdf_get_value() passing "imagewidth" as the second parameter and the image as the third parameter.
pdf_get_image_height()	pdf_get_value() passing "imageheight" as the second parameter and the image as the third parameter.

PDFlib 3.x Installation Hints

When using version 3.x of PDFlib, you should configure PDFlib with the option `--enable-shared-pdflib`.

Issues with older versions of PDFlib

Any version of PHP 4 after March 9, 2000 does not support versions of PDFlib older than 3.0.

PDFlib 3.0 or greater is supported by PHP 3.0.19 and later.

Examples

Most of the functions are fairly easy to use. The most difficult part is probably creating a very simple PDF document at all. The following example should help to get started. It creates `test.pdf` with one page. The page contains the text "Times Roman outlined" in an outlined, 30pt font. The text is also underlined.

Esempio 1. Creating a PDF document with PDFlib

```
<?php
$pdf = pdf_new();
pdf_open_file($pdf, "test.pdf");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Test for PHP wrapper of PDFlib 2.0");
pdf_set_info($pdf, "Creator", "See Author");
pdf_set_info($pdf, "Subject", "Testing");
pdf_begin_page($pdf, 595, 842);
pdf_add_outline($pdf, "Page 1");
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "Times Roman outlined", 50, 750);
pdf_moveto($pdf, 50, 740);
pdf_lineto($pdf, 330, 740);
pdf_stroke($pdf);
pdf_end_page($pdf);
pdf_close($pdf);
pdf_delete($pdf);
```

```
echo "<A HREF=getpdf.php>finished</A>";
?>
```

The script `getpdf.php` just returns the pdf document.

```
<?php
$len = filesize($filename);
header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=foo.pdf");
readfile($filename);
?>
```

The PDFlib distribution contains a more complex example which creates a page with an analog clock. Here we use the in memory creation feature of PDFlib to alleviate the need to use temporary files. The example, converted to PHP from the PDFlib example, is as follows: (The same example is available in the CLibPDF documentation.)

Esempio 2. pdfclock example from PDFlib distribution

```
<?php
$radius = 200;
$margin = 20;
$pagecount = 10;

$pdf = pdf_new();

if (!pdf_open_file($pdf, "")) {
    print error;
    exit;
};

pdf_set_parameter($pdf, "warning", "true");

pdf_set_info($pdf, "Creator", "pdf_clock.php");
pdf_set_info($pdf, "Author", "Uwe Steinmann");
pdf_set_info($pdf, "Title", "Analog Clock");

while($pagecount-- > 0) {
    pdf_begin_page($pdf, 2 * ($radius + $margin), 2 * ($radius + $margin));

    pdf_set_parameter($pdf, "transition", "wipe");
    pdf_set_value($pdf, "duration", 0.5);

    pdf_translate($pdf, $radius + $margin, $radius + $margin);
    pdf_save($pdf);
    pdf_setrgbcolor($pdf, 0.0, 0.0, 1.0);
```

```
/* minute strokes */
pdf_setlinewidth($pdf, 2.0);
for ($alpha = 0; $alpha < 360; $alpha += 6) {
    pdf_rotate($pdf, 6.0);
    pdf_moveto($pdf, $radius, 0.0);
    pdf_lineto($pdf, $radius-$margin/3, 0.0);
    pdf_stroke($pdf);
}

pdf_restore($pdf);
pdf_save($pdf);

/* 5 minute strokes */
pdf_setlinewidth($pdf, 3.0);
for ($alpha = 0; $alpha < 360; $alpha += 30) {
    pdf_rotate($pdf, 30.0);
    pdf_moveto($pdf, $radius, 0.0);
    pdf_lineto($pdf, $radius-$margin, 0.0);
    pdf_stroke($pdf);
}

$ltime = getdate();

/* draw hour hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['minutes']/60.0)+$ltime['hours']-3.0)*30.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius/2, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw minute hand */
pdf_save($pdf);
pdf_rotate($pdf,-(($ltime['seconds']/60.0)+$ltime['minutes']-15.0)*6.0);
pdf_moveto($pdf, -$radius/10, -$radius/20);
pdf_lineto($pdf, $radius * 0.8, 0.0);
pdf_lineto($pdf, -$radius/10, $radius/20);
pdf_closepath($pdf);
pdf_fill($pdf);
pdf_restore($pdf);

/* draw second hand */
pdf_setrgbcolor($pdf, 1.0, 0.0, 0.0);
pdf_setlinewidth($pdf, 2);
pdf_save($pdf);
pdf_rotate($pdf, -(($ltime['seconds'] - 15.0) * 6.0));
pdf_moveto($pdf, -$radius/5, 0.0);
pdf_lineto($pdf, $radius, 0.0);
pdf_stroke($pdf);
pdf_restore($pdf);

/* draw little circle at center */
pdf_circle($pdf, 0, 0, $radius/30);
pdf_fill($pdf);
```

```
    pdf_restore($pdf);

    pdf_end_page($pdf);

    # to see some difference
    sleep(1);
}

pdf_close($pdf);

$buf = pdf_get_buffer($pdf);
$len = strlen($buf);

header("Content-type: application/pdf");
header("Content-Length: $len");
header("Content-Disposition: inline; filename=foo.pdf");
print $buf;

pdf_delete($pdf);
?>
```

pdf_add_annotation (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Deprecated: Adds annotation

pdf_add_outline() is replaced by pdf_add_note()

See also pdf_add_note().

pdf_add_bookmark (PHP 4)

Adds bookmark for current page

int **pdf_add_bookmark** (int pdf object, string text [, int parent [, int open]]) \linebreak

Add a nested bookmark under `parent`, or a new top-level bookmark if `parent` = 0. Returns a bookmark descriptor which may be used as parent for subsequent nested bookmarks. If open = 1, child bookmarks will be folded out, and invisible if open = 0.

pdf_add_launchlink (PHP 4 >= 4.0.5)

Add a launch annotation for current page

int **pdf_add_launchlink** (int pdf object, float llx, float lly, float urx, float ury, string filename) \linebreak

Add a launch annotation (to a target of arbitrary file type).

pdf_add_locallink (PHP 4 >= 4.0.5)

Add a link annotation for current page

int **pdf_add_locallink** (int pdf object, float llx, float lly, float urx, float ury, int page, string dest) \linebreak

Add a link annotation to a target within the current PDF file.

pdf_add_note (PHP 4 >= 4.0.5)

Add a note annotation for current page

int **pdf_add_note** (int pdf object, float llx, float lly, float urx, float ury, string contents, string title, string icon, int open) \linebreak

Add a note annotation. icon is one of of "comment, "insert", "note", "paragraph", "newparagraph", "key", or "help".

pdf_add_outline (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Adds bookmark for current page

Deprecated.

See pdf_add_bookmark().

pdf_add_pdflink (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Adds file link annotation for current page

int **pdf_add_pdflink** (int pdf object, float llx, float lly, float urx, float ury, string filename, int page, string dest) \linebreak

Add a file link annotation (to a PDF target).

pdf_add_thumbnail (PHP 4 >= 4.0.5)

Adds thumbnail for current page

int **pdf_add_thumbnail** (int pdf object, int image) \linebreak

Add an existing image as thumbnail for the current page.

pdf_add_weblink (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Adds weblink for current page

int **pdf_add_weblink** (int pdf object, float llx, float lly, float urx, float ury, string url) \linebreak

Add a weblink annotation to a target URL on the Web.

pdf_arc (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Draws an arc (counterclockwise)

void **pdf_arc** (resource pdf object, float x, float y, float r, float alpha, float beta) \linebreak

Draw a counterclockwise circular arc from alpha to beta degrees

See also: pdf_arcn()

pdf_arcn (PHP 4 >= 4.0.5)

Draws an arc (clockwise)

void **pdf_arc** (resource pdf object, float x, float y, float r, float alpha, float beta) \linebreak

Draw a clockwise circular arc from alpha to beta degrees

See also: pdf_arc()

pdf_attach_file (PHP 4 >= 4.0.5)

Adds a file attachement for current page

int **pdf_attach_file** (int pdf object, float llx, float lly, float urx, float ury, string filename, string description, string author, string mimetype, string icon) \linebreak

Add a file attachment annotation. icon is one of "graph, "paperclip", "pushpin", or "tag".

pdf_begin_page (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Starts new page

void **pdf_begin_page** (int pdf object, float width, float height) \linebreak

Add a new page to the document. The *width* and *height* are specified in points, which are 1/72 of an inch.

Tabella 1. Common Page Sizes in Points

name	size
A0	2380✗3368
A1	1684✗2380
A2	1190✗1684
A3	842✗1190
A4	595✗842
A5	421✗595
A6	297✗421
B5	501✗709
letter (8.5"✗11")	612✗792
legal (8.5"✗14")	612✗1008
ledger (17"✗11")	1224✗792
11"✗17"	792✗1224

pdf_begin_pattern (PHP 4 >= 4.0.5)

Starts new pattern

int **pdf_begin_pattern** (int pdf object, float width, float height, float xstep, float ystep, int painttype) \linebreak

Starts a new pattern definition and returns a pattern handle. `width`, and `height` define the bounding box for the pattern. `xstep` and `ystep` give the repeated pattern offsets. `painttype`=1 means that the pattern has its own colour settings whereas a value of 2 indicates that the current colour is used when the pattern is applied.

pdf_begin_template (PHP 4 >= 4.0.5)

Starts new template

void **pdf_begin_template** (int pdf object, float width, float height) \linebreak

Start a new template definition.

pdf_circle (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Draws a circle

void **pdf_circle** (int pdf object, float x, float y, float r) \linebreak

Draw a circle with center (x, y) and radius r.

pdf_clip (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Clips to current path

void **pdf_clip** (int pdf object) \linebreak

Use the current path as clipping path.

pdf_close (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Closes a pdf object

void **pdf_close** (int pdf object) \linebreak

Close the generated PDF file, and free all document-related resources.

pdf_close_image (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Closes an image

void **pdf_close_image** (int pdf object, int image) \linebreak

Close an `image` retrieved with one of the **pdf_open_image*()** functions.

pdf_close_pdi (PHP 4 >= 4.0.5)

Close the input PDF document

void **pdf_close_pdi** (int pdf object, int dochandle) \linebreak

Close all open page handles, and close the input PDF document.

pdf_close_pdi_page (PHP 4 >= 4.0.5)

Close the page handle

void **pdf_close_pdi_page** (int pdf object, int pagehandle) \linebreak

Close the page handle, and free all page-related resources.

pdf_closepath (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Closes path

void **pdf_closepath** (int pdf object) \linebreak

Close the current path.

pdf_closepath_fill_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Closes, fills and strokes current path

void **pdf_closepath_fill_stroke** (int pdf object) \linebreak

Close the path, fill, and stroke it.

pdf_closepath_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Closes path and draws line along path

void **pdf_closepath_stroke** (int pdf object) \linebreak

Close the path, and stroke it.

pdf_concat (PHP 4 >= 4.0.5)

Concatenate a matrix to the CTM

void **pdf_concat** (int pdf object, float a, float b, float c, float d, float e, float f) \linebreak

Concatenate a matrix to the CTM.

pdf_continue_text (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Outputs text in next line

void **pdf_continue_text** (int pdf object, string text) \linebreak

Print text at the next line. The spacing between lines is determined by the `leading` parameter.

pdf_curveto (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Draws a curve

void **pdf_curveto** (int pdf object, float x1, float y1, float x2, float y2, float x3, float y3) \linebreak

Draw a Bezier curve from the current point, using 3 more control points.

pdf_delete (PHP 4 >= 4.0.5)

Deletes a PDF object

void **pdf_delete** (int pdf object) \linebreak

Delete the PDF object, and free all internal resources.

pdf_end_page (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Ends a page

void **pdf_end_page** (int pdf object) \linebreak

Finish the page.

pdf_end_pattern (PHP 4 >= 4.0.5)

Finish pattern

void **pdf_end_pattern** (int pdf object) \linebreak

Finish the pattern definition.

pdf_end_template (PHP 4 >= 4.0.5)

Finish template

void **pdf_end_template** (int pdf object) \linebreak

Finish the template definition.

pdf_endpath (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Ends current path

Deprecated, use one of the stroke, fill, or clip functions instead.

pdf_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Fills current path

void **pdf_fill_stroke** (int pdf object) \linebreak

Fill the interior of the path with the current fill color.

pdf_fill_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Fills and strokes current path

void **pdf_fill_stroke** (int pdf object) \linebreak

Fill and stroke the path with the current fill and stroke color.

pdf_findfont (PHP 4 >= 4.0.5)

Prepare font for later use with pdf_setfont().

int **pdf_findfont** (int pdf object, string fontname, string encoding, int embed) \linebreak

Prepare a font for later use with pdf_setfont(). The metrics will be loaded, and if embed is nonzero, the font file will be checked, but not yet used. `encoding` is one of "builtin", "macroman", "winansi", "host", or a user-defined encoding name, or the name of a CMap.

pdf_findfont() returns a font handle or `FALSE` on error.

Esempio 1. pdf_findfont() example

```
<?php

$font = pdf_findfont($pdf, "Times New Roman", "winansi", 1);
if ($font) {
    pdf_setfont($pdf, $font, 10);
}

?>
```

pdf_get_buffer (PHP 4 >= 4.0.5)

Fetch the buffer containig the generated PDF data.

string **pdf_get_buffer** (int pdf object) \linebreak

Get the contents of the PDF output buffer. The result must be used by the client before calling any other PDFlib function.

pdf_get_font (PHP 4 >= 4.0.0)

Deprecated: font handling

Deprecated.

See pdf_get_value().

pdf_get_fontname (PHP 4 >= 4.0.0)

Deprecated: font handling

Deprecated.

See pdf_get_parameter().

pdf_get_fontsize (PHP 4 >= 4.0.0)

Deprecated: font handling

Deprecated.

See pdf_get_value().

pdf_get_image_height (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Returns height of an image

string **pdf_get_image_height** (int pdf object, int image) \linebreak

pdf_get_image_height() is deprecated, use pdf_get_value() instead.

pdf_get_image_width (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Returns width of an image

string **pdf_get_image_width** (int pdf object, int image) \linebreak

The **pdf_get_image_width()** is deprecated, use pdf_get_value() instead.

pdf_get_majorversion (PHP 4 >= 4.2.0)

Returns the major version number of the PDFlib

int **pdf_get_majorversion** (void) \linebreak

Returns the major version number of the PDFlib.

pdf_get_minorversion (PHP 4 >= 4.2.0)

Returns the minor version number of the PDFlib

int **pdf_get_majorversion** (void) \linebreak

Returns the minor version number of the PDFlib.

pdf_get_parameter (PHP 4)

Gets certain parameters

string **pdf_get_parameter** (int pdf object, string key [, float modifier]) \linebreak

Get the contents of some PDFlib parameter with string type.

pdf_get_pdi_parameter (PHP 4 >= 4.0.5)

Get some PDI string parameters

string **pdf_get_pdi_parameter** (int pdf object, string key, int doc, int page, int index) \linebreak

Get the contents of some PDI document parameter with string type.

pdf_get_pdi_value (PHP 4 >= 4.0.5)

Gets some PDI numerical parameters

string **pdf_get_pdi_value** (int pdf object, string key, int doc, int page, int index) \linebreak

Get the contents of some PDI document parameter with numerical type.

pdf_get_value (PHP 4)

Gets certain numerical value

float **pdf_get_value** (int pdf object, string key [, float modifier]) \linebreak

Get the contents of some PDFlib parameter with float type.

pdf_initgraphics (PHP 4 >= 4.0.5)

Resets graphic state

void **pdf_initgraphics** (int pdf object) \linebreak

Reset all implicit color and graphics state parameters to their defaults.

pdf_lineto (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Draws a line

void **pdf_lineto** (int pdf object, float x, float y) \linebreak

Draw a line from the current point to (x, y).

pdf_makespotcolor (PHP 4 >= 4.0.5)

Makes a spotcolor

void **pdf_makespotcolor** (int pdf object, string spotname) \linebreak

Make a named spot color from the current color.

pdf_moveto (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets current point

void **pdf_moveto** (int pdf object, float x, float y) \linebreak

Set the current point.

> **Nota:** The current point for graphics and the current text output position are maintained separately. See pdf_set_text_pos() to set the text output position.

pdf_new (PHP 4 >= 4.0.5)

Creates a new pdf object

int **pdf_new** ( ) \linebreak

Create a new PDF object, using default error handling and memory management.

pdf_open (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Open a new pdf object

pdf_open() is deprecated, use pdf_new() plus pdf_open_file() instead.

See also pdf_new(), pdf_open_file().

pdf_open_CCITT (PHP 4 >= 4.0.5)

Opens a new image file with raw CCITT data

int **pdf_open_CCITT** (int pdf object, string filename, int width, int height, int BitReverse, int k, int Blackls1) \linebreak

Open a raw CCITT image.

pdf_open_file (PHP 4 >= 4.0.5)

Opens a new pdf object

int **pdf_open_file** (int pdf object [, string filename]) \linebreak

Create a new PDF file using the supplied file name. If *filename* is empty the PDF document will be generated in memory instead of on file. The result must be fetched by the client with the pdf_get_buffer() function.

The following example shows how to create a pdf document in memory and how to output it correctly.

Esempio 1. Creating a PDF document in memory

```
<?php

$pdf = pdf_new();

pdf_open_file($pdf);
pdf_begin_page($pdf, 595, 842);
pdf_set_font($pdf, "Times-Roman", 30, "host");
pdf_set_value($pdf, "textrendering", 1);
pdf_show_xy($pdf, "A PDF document created in memory!", 50, 750);
pdf_end_page($pdf);
pdf_close($pdf);

$data = pdf_get_buffer($pdf);

header("Content-type: application/pdf");
header("Content-disposition: inline; filename=test.pdf");
header("Content-length: " . strlen($data));

echo $data;

?>
```

pdf_open_gif (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Deprecated: Opens a GIF image

Deprecated.

See pdf_open_image(),

pdf_open_image (PHP 4 >= 4.0.5)

Versatile function for images

int **pdf_open_image** (int PDF-document, string imagetype, string source, string data, long length, int width, int height, int components, int bpc, string params) \linebreak

Use image data from a variety of data sources. Supported types are "jpeg", "ccitt", "raw". Supported sources are "memory", "fileref", "url". len is only used for type="raw", params is only used for type="ccitt".

pdf_open_image_file (PHP 3 CVS only, PHP 4 >= 4.0.0)

Reads an image from a file

int **pdf_open_image_file** (int PDF-document, string imagetype, string filename [, string stringparam [, string intparam]]) \linebreak

Open an image file. Supported types are "jpeg", "tiff", "gif", and "png". *`stringparam`* is either "", "mask", "masked", or "page". *`intparam`*is either 0, the image id of the applied mask, or the page.

pdf_open_jpeg (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Deprecated: Opens a JPEG image

Deprecated.

See also pdf_open_image(),

pdf_open_memory_image (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Opens an image created with PHP's image functions

int **pdf_open_memory_image** (int pdf object, int image) \linebreak

The **pdf_open_memory_image()** function takes an image created with the PHP's image functions and makes it available for the pdf object. The function returns a pdf image identifier.

Esempio 1. Including a memory image

```
<?php
$im = ImageCreate(100, 100);
$col = ImageColorAllocate($im, 80, 45, 190);
ImageFill($im, 10, 10, $col);
$pim = pdf_open_memory_image($pdf, $im);
ImageDestroy($im);
pdf_place_image($pdf, $pim, 100, 100, 1);
```

```
pdf_close_image($pdf, $pim);
?>
```

See also pdf_close_image(), pdf_place_image().

pdf_open_pdi (PHP 4 >= 4.0.5)

Opens a PDF file

int **pdf_open_pdi** (int pdf object, string filename, string stringparam, int intparam) \linebreak

Open an existing PDF document for later use.

pdf_open_pdi_page (PHP 4 >= 4.0.5)

Prepare a page

int **pdf_open_pdi_page** (int pdf object, int dochandle, int pagenumber, string pagelabel) \linebreak

Prepare a page for later use with pdf_place_image()

pdf_open_png (PHP 4 >= 4.0.0)

Deprecated: Opens a PNG image

Deprecated.

See pdf_open_image().

pdf_open_tiff (PHP 4 >= 4.0.0)

Deprecated: Opens a TIFF image

int **pdf_open_tiff** (int PDF-document, string filename) \linebreak

Deprecated.

See also pdf_open_image(),

pdf_place_image (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Places an image on the page

void **pdf_place_image** (int pdf object, int image, float x, float y, float scale) \linebreak

Place an image with the lower left corner at (x, y), and scale it.

pdf_place_pdi_page (PHP 4 >= 4.0.6)

Places an image on the page

void **pdf_place_pdi_page** (int pdf object, int page, float x, float y, float sx, float sy) \linebreak

Place a PDF page with the lower left corner at (x, y), and scale it.

pdf_rect (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Draws a rectangle

void **pdf_rect** (int pdf object, float x, float y, float width, float height) \linebreak

Draw a rectangle at lower left (x, y) with width and height.

pdf_restore (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restores formerly saved environment

void **pdf_restore** (int pdf object) \linebreak

Restore the most recently saved graphics state.

pdf_rotate (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets rotation

void **pdf_rotate** (int pdf object, float phi) \linebreak

Rotate the coordinate system by phi degrees.

pdf_save (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Saves the current environment

void **pdf_save** (int pdf object) \linebreak

Save the current graphics state.

pdf_scale (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets scaling

void **pdf_scale** (int pdf object, float x-scale, float y-scale) \linebreak

Scale the coordinate system.

pdf_set_border_color (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Sets color of border around links and annotations

void **pdf_set_border_color** (int pdf object, float red, float green, float blue) \linebreak

Set the border color for all kinds of annotations.

pdf_set_border_dash (PHP 4)

Sets dash style of border around links and annotations

void **pdf_set_border_dash** (int pdf object, float black, float white) \linebreak

Set the border dash style for all kinds of annotations. See pdf_setdash().

pdf_set_border_style (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Sets style of border around links and annotations

void **pdf_set_border_style** (int pdf object, string style, float width) \linebreak

Set the border style for all kinds of annotations. `style` is "solid" or "dashed".

pdf_set_char_spacing (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Sets character spacing

Deprecated.

See also pdf_set_value(),

pdf_set_duration (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Sets duration between pages

Deprecated.

See pdf_set_value().

pdf_set_font (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Selects a font face and size

Deprecated. You should use pdf_findfont() plus pdf_setfont() instead.

See pdf_findfont(), pdf_setfont().

pdf_set_horiz_scaling (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets horizontal scaling of text

void **pdf_set_horiz_scaling** (int pdf object, float scale) \linebreak

Deprecated.

See also pdf_set_value(),

pdf_set_info (PHP 4)

Fills a field of the document information

void **pdf_set_info** (int pdf object, string key, string value) \linebreak

Fill document information field key with value. *key* is one of "Subject", "Title", "Creator", "Author", "Keywords", or a user-defined key.

pdf_set_info_author (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Fills the author field of the document

bool **pdf_set_info_author** (int pdfdoc, string author) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_creator (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Fills the creator field of the document

bool **pdf_set_info_creator** (int pdfdoc, string creator) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_keywords (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Fills the keywords field of the document

bool **pdf_set_info_keywords** (int pdfdoc, string keywords) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_subject (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Fills the subject field of the document

bool **pdf_set_info_subject** (int pdfdoc, string subject) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_info_title (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Fills the title field of the document

bool **pdf_set_info_title** (int pdfdoc, string title) \linebreak

This function is deprecate, use pdf_set_info() instead.

pdf_set_leading (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Sets distance between text lines

Deprecated.

See also pdf_set_value(),

pdf_set_parameter (PHP 4 >= 4.0.0)

Sets certain parameters

void **pdf_set_parameter** (int pdf object, string key, string value) \linebreak

Set some PDFlib parameter with string type.

pdf_set_text_pos (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets text position

void **pdf_set_text_pos** (int pdf object, float x, float y) \linebreak

Set the text output position.

pdf_set_text_rendering (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Determines how text is rendered

Deprecated.

See pdf_set_value(),

pdf_set_text_rise (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Deprecated: Sets the text rise

Deprecated.

See pdf_set_value(),

pdf_set_text_matrix (PHP 3>= 3.0.6)

Deprecated: Sets the text matrix

See **pdf_set_paramter()**.

pdf_set_value (PHP 4)

Sets certain numerical value

void **pdf_set_value** (int pdf object, string key, float value) \linebreak

Set the value of some PDFlib parameter with float type.

pdf_set_word_spacing (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Depriciated: Sets spacing between words

Deprecated.

See also pdf_set_value(),

pdf_setcolor (PHP 4 >= 4.0.5)

Sets fill and stroke color

void **pdf_setcolor** (int pdf object, string type, string colorspace, float c1 [, float c2 [, float c3 [, float c4]]]) \linebreak

Set the current color space and color. The parameter *type* can be "fill", "stroke", or "both" to specify that the color is set for filling, stroking or both filling and stroking. The parameter *colorspace* can be `gray`, `rgb`, `cmyk`, `spot` or `pattern`. The parameters *c1*, *c2*, *c3* and *c4* represent the color components for the color space specified by *colorspace*. Except as otherwise noted, the color components are floating-point values that range from 0 to 1.

For `gray` only *c1* is used.

For `rgb` parameters *c1*, *c2*, and *c3* specify the red, green and blue values respectively.

```
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "rgb", 1, 1, 1);
```

For `cmyk`, parameters *c1*, *c2*, *c3*, and *c4* are the cyan, magenta, yellow and black values, respectively.

```
// Set fill and stroke colors to white.
pdf_setcolor($pdf, "both", "cmyk", 0, 0, 0, 1);
```

For `spot`, *c1* should be a spot color handles returned by pdf_makespotcolor() and *c2* is a tint value between 0 and 1.

For `pattern`, *c1* should be a pattern handle returned by pdf_begin_pattern().

pdf_setdash (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets dash pattern

void **pdf_setdash** (int pdf object, float b, float w) \linebreak

Set the current dash pattern to *b* black and *w* white units.

pdf_setflat (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets flatness

void **pdf_setflat** (int pdf object, float flatness) \linebreak

Set the flatness to a value between 0 and 100 inclusive.

pdf_setfont (PHP 4 >= 4.0.5)

Set the current font

void **pdf_setfont** (int pdf object, int font, float size) \linebreak

Set the current font in the given size, using a `font` handle returned by pdf_findfont()

See Also: pdf_findfont().

pdf_setgray (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets drawing and filling color to gray value

void **pdf_setgray** (int pdf object, float gray) \linebreak

Set the current fill and stroke color.

> **Nota:** PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setgray_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets filling color to gray value

void **pdf_setgray_fill** (int pdf object, float gray) \linebreak

Set the current fill color to a gray value between 0 and 1 inclusive.

> **Nota:** PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setgray_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets drawing color to gray value

void **pdf_setgray_stroke** (int pdf object, float gray) \linebreak

Set the current stroke color to a gray value between 0 and 1 inclusive

Nota: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setlinecap (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets linecap parameter

void **pdf_setlinecap** (int pdf object, int linecap) \linebreak

Set the *`linecap`* parameter to a value between 0 and 2 inclusive.

pdf_setlinejoin (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets linejoin parameter

void **pdf_setlinejoin** (int pdf object, long linejoin) \linebreak

Set the line join parameter to a value between 0 and 2 inclusive.

pdf_setlinewidth (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets line width

void **pdf_setlinewidth** (int pdf object, float width) \linebreak

Set the current linewidth to width.

pdf_setmatrix (PHP 4 >= 4.0.5)

Sets current transformation matrix

void **pdf_setmatrix** (int pdf object, float a, float b, float c, float d, float e, float f) \linebreak

Explicitly set the current transformation matrix.

pdf_setmiterlimit (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets miter limit

void **pdf_setmiterlimit** (int pdf object, float miter) \linebreak

Set the miter limit to a value greater than or equal to 1.

pdf_setpolydash (PHP 4 >= 4.0.5)

Sets complicated dash pattern

void **pdf_setpolydash** (int pdf object, float * dasharray) \linebreak

Set a more complicated dash pattern defined by an array.

pdf_setrgbcolor (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets drawing and filling color to rgb color value

void **pdf_setrgbcolor** (int pdf object, float red value, float green value, float blue value) \linebreak

Set the current fill and stroke color to the supplied RGB values.

Nota: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setrgbcolor_fill (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets filling color to rgb color value

void **pdf_setrgbcolor_fill** (int pdf object, float red value, float green value, float blue value) \linebreak

Set the current fill color to the supplied RGB values.

Nota: PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_setrgbcolor_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets drawing color to rgb color value

void **pdf_setrgbcolor_stroke** (int pdf object, float red value, float green value, float blue value) \linebreak

Set the current stroke color to the supplied RGB values.

> **Nota:** PDFlib V4.0: Deprecated, use pdf_setcolor() instead.

pdf_show (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Output text at current position

void **pdf_show** (int pdf object, string text) \linebreak

Print text in the current font and size at the current position.

pdf_show_boxed (PHP 4 >= 4.0.0)

Output text in a box

int **pdf_show_boxed** (int pdf object, string text, float left, float top, float width, float height, string hmode [, string feature]) \linebreak

Format text in the current font and size into the supplied text box according to the requested formatting mode, which must be one of "left", "right", "center", "justify", or "fulljustify". If width and height are 0, only a single line is placed at the point (left, top) in the requested mode.

Returns the number of characters that did not fit in the specified box. Returns 0 if all characters fit or the *width* and *height* parameters were set to 0 for single-line formattting.

pdf_show_xy (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Output text at given position

void **pdf_show_xy** (int pdf object, string text, float x, float y) \linebreak

Print text in the current font at (x, y).

pdf_skew (PHP 4 >= 4.0.0)

Skews the coordinate system

void **pdf_skew** (int pdf object, float alpha, float beta) \linebreak

Skew the coordinate system in x and y direction by alpha and beta degrees.

pdf_stringwidth (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Returns width of text using current font

float **pdf_stringwidth** (int pdf object, string text [, int font [, float size]]) \linebreak

Returns the width of *text* using the last font set by pdf_setfont(). If the optional parameters *font* and *size* are specified, the width will be calculated using that font and size instead. Please note that *font* is a font handle returned by pdf_findfont().

> **Nota:** Both the *font* and *size* parameters must used together.

See Also: pdf_setfont() and pdf_findfont().

pdf_stroke (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Draws line along path

void **pdf_stroke** (int pdf object) \linebreak

Stroke the path with the current color and line width, and clear it.

pdf_translate (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets origin of coordinate system

void **pdf_translate** (int pdf object, float tx, float ty) \linebreak

Translate the origin of the coordinate system.

LXXVI. Verisign Payflow Pro functions

This extension allows you to process credit cards and other financial transactions using Verisign Payment Services, formerly known as Signio (http://www.verisign.com/products/payflow/pro/index.html).

These functions are only available if PHP has been compiled with the `--with-pfpro[=DIR]` option. You will require the appropriate SDK for your platform, which may be downloaded from within the manager interface (https://manager.verisign.com/) once you have registered. If you are going to use this extension in an SSL-enabled webserver or with other SSL components (such as the CURL+SSL extension) you MUST get the beta SDK.

Once you have downloaded the SDK you should copy the files from the `lib` directory of the distribution. Copy the header file `pfpro.h` to `/usr/local/include` and the library file `libpfpro.so` to `/usr/local/lib`.

When using these functions, you may omit calls to pfpro_init() and pfpro_cleanup() as this extension will do so automatically if required. However the functions are still available in case you are processing a number of transactions and require fine control over the library. You may perform any number of transactions using pfpro_process() between the two.

These functions have been added in PHP 4.0.2.

> **Nota:** These functions only provide a link to Verisign Payment Services. Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

pfpro_cleanup (PHP 4 >= 4.0.2)

Shuts down the Payflow Pro library

void **pfpro_cleanup** (void) \linebreak

pfpro_cleanup() is used to shutdown the Payflow Pro library cleanly. It should be called after you have processed any transactions and before the end of your script. However you may omit this call, in which case this extension will automatically call **pfpro_cleanup()** after your script terminates.

See also pfpro_init().

pfpro_init (PHP 4 >= 4.0.2)

Initialises the Payflow Pro library

void **pfpro_init** (void) \linebreak

pfpro_init() is used to initialise the Payflow Pro library. You may omit this call, in which case this extension will automatically call **pfpro_init()** before the first transaction.

See also pfpro_cleanup().

pfpro_process (PHP 4 >= 4.0.2)

Process a transaction with Payflow Pro

array **pfpro_process** (array parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]]) \linebreak

Returns: An associative array containing the response

pfpro_process() processes a transaction with Payflow Pro. The first parameter is an associative array containing keys and values that will be encoded and passed to the processor.

The second parameter is optional and specifies the host to connect to. By default this is "test.signio.com", so you will certainly want to change this to "connect.signio.com" in order to process live transactions.

The third parameter specifies the port to connect on. It defaults to 443, the standard SSL port.

The fourth parameter specifies the timeout to be used, in seconds. This defaults to 30 seconds. Note that this timeout appears to only begin once a link to the processor has been established and so your script could potentially continue for a very long time in the event of DNS or network problems.

The fifth parameter, if required, specifies the hostname of your SSL proxy. The sixth parameter specifies the port to use.

The seventh and eighth parameters specify the logon identity and password to use on the proxy.

The function returns an associative array of the keys and values in the response.

> **Nota:** Be sure to read the Payflow Pro Developers Guide for full details of the required parameters.

Esempio 1. Payflow Pro example

```
<?php

pfpro_init();

$transaction = array(USER => ’mylogin’,
     PWD => ’mypassword’,
     TRXTYPE => ’S’,
     TENDER => ’C’,
     AMT => 1.50,
     ACCT => ’4111111111111111’,
     EXPDATE => ’0904’
     );

$response = pfpro_process($transaction);

if (!$response) {
  die("Couldn’t establish link to Verisign.\n");
}

echo "Verisign response code was ".$response[RESULT];
echo ", which means: ".$response[RESPMSG]."\n";

echo "\nThe transaction request: ";
print_r($transaction);

echo "\nThe response: ";
print_r($response);

pfpro_cleanup();

?>
```

pfpro_process_raw (PHP 4 >= 4.0.2)

Process a raw transaction with Payflow Pro

string **pfpro_process_raw** (string parameters [, string address [, int port [, int timeout [, string proxy address [, int proxy port [, string proxy logon [, string proxy password]]]]]]]) \linebreak

Returns: A string containing the response.

pfpro_process_raw() processes a raw transaction string with Payflow Pro. You should really use pfpro_process() instead, as the encoding rules of these transactions are non-standard.

The first parameter in this case is a string containing the raw transaction request. All other parameters are the same as with pfpro_process(). The return value is a string containing the raw response.

Nota: Be sure to read the Payflow Pro Developers Guide for full details of the required parameters and encoding rules. You would be well advised to use pfpro_process() instead.

Esempio 1. Payflow Pro raw example

```
<?php

pfpro_init();

$response = pfpro_process("USER=mylogin&PWD[5]=m&ndy&TRXTYPE=S&TENDER=C&AMT=1.50&ACCT=41

if (!$response) {
  die("Couldn’t establish link to Verisign.\n");
}

echo "Verisign raw response was ".$response;

pfpro_cleanup();

?>
```

pfpro_version (PHP 4 >= 4.0.2)

Returns the version of the Payflow Pro software

string **pfpro_version** (void) \linebreak

pfpro_version() returns the version string of the Payflow Pro library. At the time of writing, this was L211.

LXXVII. PHP Options&Information

assert (PHP 4 >= 4.0.0)

Checks if assertion is FALSE

int **assert** (string|bool assertion) \linebreak

assert() will check the given *assertion* and take appropriate action if its result is FALSE.

If the *assertion* is given as a string it will be evaluated as PHP code by **assert()**. The advantages of a string *assertion* are less overhead when assertion checking is off and messages containing the *assertion* expression when an assertion fails.

Assertions should be used as a debugging feature only. You may use them for sanity-checks that test for conditions that should always be TRUE and that indicate some programming errors if not or to check for the presence of certain features like extension functions or certain system limits and features.

Assertions should not be used for normal runtime operations like input parameter checks. As a rule of thumb your code should always be able to work correctly if assertion checking is not activated.

The behavior of **assert()** may be configured by assert_options() or by .ini-settings described in that functions manual page.

The assert_options() function and/or ASSERT_CALLBACK configuration directive allow a callback function to be set to handle failed assertions.

assert() callbacks are particularly useful for building automated test suites because they allow you to easily capture the code passed to the assertion, along with information on where the assertion was made. While this information can be captured via other methods, using assertions makes it much faster and easier!

The callback function should accept three arguments. The first argument will contain the file the assertion failed in. The second argument will contain the line the assertion failed on and the third argument will contain the expression that failed (if any - literal values such as 1 or "two" will not be passed via this argument)

Esempio 1. Handle a failed assertion with a custom handler

```
<?php
// Active assert and make it quiet
assert_options (ASSERT_ACTIVE, 1);
assert_options (ASSERT_WARNING, 0);
assert_options (ASSERT_QUIET_EVAL, 1);

// Create a handler function
function my_assert_handler ($file, $line, $code) {
    echo "<hr>Assertion Failed:
        File '$file'<br>
        Line '$line'<br>
        Code '$code'<br><hr>";
}

// Set up the callback
assert_options (ASSERT_CALLBACK, 'my_assert_handler');

// Make an assertion that should fail
assert ('mysql_query ("")');
```

```
?>
```

assert_options (PHP 4 >= 4.0.0)

Set/get the various assert flags

mixed **assert_options** (int what [, mixed value]) \linebreak

Using **assert_options()** you may set the various assert() control options or just query their current settings.

Tabella 1. Assert Options

option	ini-parameter	default	description
ASSERT_ACTIVE	assert.active	1	enable assert() evaluation
ASSERT_WARNING	assert.warning	1	issue a PHP warning for each failed assertion
ASSERT_BAIL	assert.bail	0	terminate execution on failed assertions
ASSERT_QUIET_EVAL	assert.quiet_eval	0	disable error_reporting during assertion expression evaluation
ASSERT_CALLBACK	assert_callback	(`NULL`)	user function to call on failed assertions

assert_options() will return the original setting of any option or `FALSE` on errors.

dl (PHP 3, PHP 4 >= 4.0.0)

Loads a PHP extension at runtime

bool **dl** (string library) \linebreak

Loads the PHP extension given by the parameter *`library`*. The *`library`* parameter is *only* the filename of the extension to load which also depends on your platform. For example, the sockets extension (if compiled as a shared module, not the default!) would be called `sockets.so` on unix platforms whereas it is called `php_sockets.dll` on the windows platform.

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento. If the functionality of loading modules is not available (see Note) or has been disabled (either by turning it off `enable_dl` or by enabling `safe_mode` in `php.ini`) an `E_ERROR` is emitted and execution is stopped. If **dl()** fails because the specified library couldn't be loaded, in addition to `FALSE` an `E_WARNING` message is emitted.

Use extension_loaded() to test whether a given extension is already available or not. This works on both built-in extensions and dynamically loaded ones (either through `php.ini` or **dl()**).

Example:

```
if (!extension_loaded('gd')) {
    if (!dl('gd.so')) {
        exit;
    }
}
```

The directory where the extension is loaded from depends on your platform:

Windows - If not explicitly set in the `php.ini`, the extension is loaded from `c:\php4\extensions\` by default.

Unix - If not explicitly set in the `php.ini`, the default extension directory depends on

- whether PHP has been built with `--enable-debug` or not
- whether PHP has been built with (experimental) ZTS (Zend Thread Safety) support or not
- the current internal `ZEND_MODULE_API_NO` (Zend internal module API number, which is basically the date on which a major module API change happened, e.g. `20010901`)

Taking into account the above, the directory then defaults to `<php-install-directory>/lib/php/extension/<debug-or-not>-<zts-or-not>-ZEND_MODULE_API_NO`, e.g. `/usr/local/php/lib/php/extensions/debug-non-zts-20010901` or `/usr/local/php/lib/php/extensions/no-debug-zts-20010901`.

> **Nota: dl()** is *not* supported in multithreaded Web servers. Use the `extensions` statement in your `php.ini` when operating under such an environment. However, the `CGI` and `CLI` build are **not** affected !

> **Nota: dl()** is case sensitive on unix platforms.

See also Extension Loading Directives and extension_loaded().

extension_loaded (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Find out whether an extension is loaded

bool **extension_loaded** (string name) \linebreak

Returns `TRUE` if the extension identified by *`name`* is loaded, `FALSE` otherwise.

Example:

```
if (!extension_loaded('gd')) {
    if (!dl('gd.so')) {
```

```
        exit;
    }
}
```

You can see the names of various extensions by using phpinfo() or if you're usnig the `CGI` or `CLI` version of `PHP` you can use the `-m` switch to list all available extensions:

```
$ php -m
[PHP Modules]
xml
tokenizer
standard
sockets
session
posix
pcre
overload
mysql
mbstring
ctype

[Zend Modules]
```

> **Nota: extension_loaded()** uses the internal extension name to test whether a certain extension is available or not. Most internal extension names are written in lower case but there may be extension available which also use uppercase letters. Be warned that this function compares **case sensitive** !

See also phpinfo() and dl().

get_cfg_var (PHP 3, PHP 4 >= 4.0.0)

Gets the value of a PHP configuration option

string **get_cfg_var** (string varname) \linebreak

Returns the current value of the PHP configuration variable specified by *`varname`*, or `FALSE` if an error occurs.

It will not return configuration information set when the PHP was compiled, or read from an Apache configuration file (using the php3_configuration_option directives).

To check whether the system is using a configuration file, try retrieving the value of the cfg_file_path configuration setting. If this is available, a configuration file is being used.

See also ini_get().

get_current_user (PHP 3, PHP 4 >= 4.0.0)

Gets the name of the owner of the current PHP script

string **get_current_user** (void) \linebreak

Returns the name of the owner of the current PHP script.

See also getmyuid(), getmygid(), getmypid(), getmyinode(), and getlastmod().

get_defined_constants (PHP 4 >= 4.1.0)

Returns an associative array with the names of all the constants and their values

array **get_defined_constants** (void) \linebreak

This function returns the names and values of all the constants currently defined. This includes those created by extensions as well as those created with the define() function.

For example the line below

```
print_r (get_defined_constants());
```

will print a list like:

```
Array
(
    [E_ERROR] => 1
    [E_WARNING] => 2
    [E_PARSE] => 4
    [E_NOTICE] => 8
    [E_CORE_ERROR] => 16
    [E_CORE_WARNING] => 32
    [E_COMPILE_ERROR] => 64
    [E_COMPILE_WARNING] => 128
    [E_USER_ERROR] => 256
    [E_USER_WARNING] => 512
    [E_USER_NOTICE] => 1024
    [E_ALL] => 2047
    [TRUE] => 1
)
```

See also get_loaded_extensions(), get_defined_functions() and get_defined_vars().

get_extension_funcs (PHP 4 >= 4.0.0)

Returns an array with the names of the functions of a module

array **get_extension_funcs** (string module_name) \linebreak

This function returns the names of all the functions defined in the module indicated by *module_name*.

For example the lines below

```
print_r (get_extension_funcs ("xml"));
print_r (get_extension_funcs ("gd"));
```

will print a list of the functions in the modules `xml` and `gd` respectively.

See also: get_loaded_extensions()

get_included_files (PHP 4 >= 4.0.0)

Returns an array with the names of included or required files

array **get_included_files** (void) \linebreak

Returns an array of the names of all files that have been included using include(), include_once(), require() or require_once().

Files that are included or required multiple times only show up once in the returned array.

Nota: Files included using the `auto_prepend_file` configuration directive are not included in the returned array.

Esempio 1. get_included_files() Example

```
<?php

include("test1.php");
include_once("test2.php");
require("test3.php");
require_once("test4.php");

$included_files = get_included_files();

foreach($included_files as $filename) {
    echo "$filename\n";
}

?>
```

will generate the following output:

```
test1.php
test2.php
test3.php
test4.php
```

> **Nota:** In PHP 4.0.1pl2 and previous versions **get_included_files()** assumed that the required files ended in the extension `.php`; other extensions would not be returned. The array returned by **get_included_files()** was an associative array and only listed files included by include() and include_once().

See also: include(), include_once(), require(), require_once(), and get_required_files().

get_loaded_extensions (PHP 4 >= 4.0.0)

Returns an array with the names of all modules compiled and loaded

array **get_loaded_extensions** (void) \linebreak

This function returns the names of all the modules compiled and loaded in the PHP interpreter.

For example the line below

```
print_r (get_loaded_extensions());
```

will print a list like:

```
Array
(
   [0] => xml
   [1] => wddx
   [2] => standard
   [3] => session
   [4] => posix
   [5] => pgsql
   [6] => pcre
   [7] => gd
   [8] => ftp
   [9] => db
```

```
    [10] => calendar
    [11] => bcmath
)
```

See also: get_extension_funcs().

get_magic_quotes_gpc (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gets the current active configuration setting of magic quotes gpc

long **get_magic_quotes_gpc** (void) \linebreak

Returns the current active configuration setting of magic_quotes_gpc (0 for off, 1 for on).

See also get_magic_quotes_runtime() and set_magic_quotes_runtime().

get_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Gets the current active configuration setting of magic_quotes_runtime

long **get_magic_quotes_runtime** (void) \linebreak

Returns the current active configuration setting of magic_quotes_runtime (0 for off, 1 for on).

See also get_magic_quotes_gpc() and set_magic_quotes_runtime().

get_required_files (PHP 4 >= 4.0.0)

Returns an array with the names of included or required files

array **get_required_files** (void) \linebreak

As of PHP 4.0.4, this function is an alias for get_included_files()

In PHP 4.0.1pl2 and previous versions **get_required_files()** assumed that the required files ended in the extension `.php`, other extensions would not be returned. The array returned by **get_required_files()** was an associative array and only listed files included by require() and require_once().

See also: require(), require_once(), include(), include_once(), and get_included_files().

getenv (PHP 3, PHP 4 >= 4.0.0)

Gets the value of an environment variable

string **getenv** (string varname) \linebreak

Returns the value of the environment variable *varname*, or FALSE on an error.

```
$ip = getenv ("REMOTE_ADDR"); // get the ip number of the user
```

You can see a list of all the environmental variables by using phpinfo(). You can find out what many of them mean by taking a look at the CGI specification (http://hoohoo.ncsa.uiuc.edu/cgi/), specifically the page on environmental variables (http://hoohoo.ncsa.uiuc.edu/cgi/env.html).

> **Nota:** This function does not work in ISAPI mode.

See also putenv().

getlastmod (PHP 3, PHP 4 >= 4.0.0)

Gets time of last page modification

int **getlastmod** (void) \linebreak

Returns the time of the last modification of the current page. The value returned is a Unix timestamp, suitable for feeding to date(). Returns FALSE on error.

Esempio 1. getlastmod() example

```
// outputs e.g. 'Last modified: March 04 1998 20:43:59.'
echo "Last modified: " . date ("F d Y H:i:s.", getlastmod());
```

See also date(), getmyuid(), getmygid(), get_current_user(), getmyinode(), and getmypid().

getmygid (PHP 4 >= 4.1.0)

Get PHP script owner's GID

int **getmygid** (void) \linebreak

Returns the group ID of the current script, or FALSE on error.

See also getmyuid(), getmypid(), get_current_user(), getmyinode(), and getlastmod().

getmyinode (PHP 3, PHP 4 >= 4.0.0)

Gets the inode of the current script

int **getmyinode** (void) \linebreak

Returns the current script's inode, or FALSE on error.

See also getmygid(), getmyuid(), get_current_user(), getmypid(), and getlastmod().

> **Nota:** Questa funzione non è implementata su piattaforme Windows

getmypid (PHP 3, PHP 4 >= 4.0.0)

Gets PHP's process ID

int **getmypid** (void) \linebreak

Returns the current PHP process ID, or FALSE on error.

> **Attenzione**
>
> Process IDs are not unique, thus they are a weak entropy source. We recommend against relying on pids in security-dependent contexts.

See also getmygid(), getmyuid(), get_current_user(), getmyinode(), and getlastmod().

getmyuid (PHP 3, PHP 4 >= 4.0.0)

Gets PHP script owner's UID

int **getmyuid** (void) \linebreak

Returns the user ID of the current script, or FALSE on error.

See also getmygid(), getmypid(), get_current_user(), getmyinode(), and getlastmod().

getrusage (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Gets the current resource usages

array **getrusage** ([int who]) \linebreak

This is an interface to getrusage(2). It returns an associative array containing the data returned from the system call. If who is 1, getrusage will be called with RUSAGE_CHILDREN.

All entries are accessible by using their documented field names.

Esempio 1. Getrusage Example

```
$dat = getrusage();
echo $dat["ru_nswap"];         # number of swaps
echo $dat["ru_majflt"];        # number of page faults
echo $dat["ru_utime.tv_sec"];  # user time used (seconds)
echo $dat["ru_utime.tv_usec"]; # user time used (microseconds)
```

See your system's man page on getrusage(2) for more details.

ini_alter (PHP 4 >= 4.0.0)

Changes the value of a configuration option

string **ini_alter** (string varname, string newvalue) \linebreak

Changes the value of a configuration option, returns FALSE on failure, and the previous value of the configuration option on success.

> **Nota:** This is an alias of ini_set()

See also ini_get(), ini_get_all(), ini_restore(), and ini_set()

ini_get (PHP 4 >= 4.0.0)

Gets the value of a configuration option

string **ini_get** (string varname) \linebreak

Returns the value of the configuration option on success, an empty string on failure.

See also get_cfg_var(), ini_get_all(), ini_alter(), ini_restore(), and ini_set()

ini_get_all (PHP 4 >= 4.2.0)

Gets all configuration options

array **ini_get_all** ([string extension]) \linebreak

Returns all the registered configuration options as an associative array. If optional *extension* parameter is set, returns only options specific for that extension.

See also: ini_alter(), ini_restore(), ini_get(), and ini_set()

ini_restore (PHP 4 >= 4.0.0)

Restores the value of a configuration option

string **ini_restore** (string varname) \linebreak

Restores a given configuration option to its original value.

See also: ini_alter(), ini_get(), ini_get_all(), and ini_set()

ini_set (PHP 4 >= 4.0.0)

Sets the value of a configuration option

string **ini_set** (string varname, string newvalue) \linebreak

Sets the value of the given configuration option. Returns the old value on success, FALSE on failure. The configuration option will keep this new value during the script's execution, and will be restored at the script's ending.

Not all the available options can be changed using **ini_set()**. Below is a table with a list of all PHP options (as of PHP 4.2.0), indicating which ones can be changed/set and at what level.

Tabella 1. Configuration options

Name	Default	Changeable
com.allow_dcom	"0"	PHP_INI_SYSTEM
com.autoregister_typelib	"0"	PHP_INI_SYSTEM
com.autoregister_verbose	"0"	PHP_INI_SYSTEM
com.autoregister_casesensitive	"1"	PHP_INI_SYSTEM
com.typelib_file	""	PHP_INI_SYSTEM
crack.default_dictionary	NULL	PHP_INI_SYSTEM
exif.encode_unicode	"ISO-8859-15"	PHP_INI_ALL
exif.decode_unicode_motorola	"UCS-2BE"	PHP_INI_ALL
exif.decode_unicode_intel	"UCS-2LE"	PHP_INI_ALL
exif.encode_jis	""	PHP_INI_ALL
exif.decode_jis_motorola	"JIS"	PHP_INI_ALL
exif.decode_jis_intel	"JIS"	PHP_INI_ALL
fbsql.allow_persistent	"1"	PHP_INI_SYSTEM
fbsql.generate_warnings	"0"	PHP_INI_SYSTEM
fbsql.autocommit	"1"	PHP_INI_SYSTEM
fbsql.max_persistent	"-1"	PHP_INI_SYSTEM
fbsql.max_links	"128"	PHP_INI_SYSTEM
fbsql.max_connections	"128"	PHP_INI_SYSTEM
fbsql.max_results	"128"	PHP_INI_SYSTEM
fbsql.batchSize	"1000"	PHP_INI_SYSTEM
fbsql.default_host	NULL	PHP_INI_SYSTEM

Name	Default	Changeable
fbsql.default_user	"_SYSTEM"	PHP_INI_SYSTEM
fbsql.default_password	""	PHP_INI_SYSTEM
fbsql.default_database	""	PHP_INI_SYSTEM
fbsql.default_database_password	""	PHP_INI_SYSTEM
hwapi.allow_persistent	"0"	PHP_INI_SYSTEM
hyerwave.allow_persistent	"0"	PHP_INI_SYSTEM
hyperwave.default_port	"418"	PHP_INI_ALL
iconv.input_encoding	ICONV_INPUT_ENCODING	PHP_INI_ALL
iconv.output_encoding	ICONV_OUTPUT_ENCODING	PHP_INI_ALL
iconv.internal_encoding	ICONV_INTERNAL_ENCODING	PHP_INI_ALL
ifx.allow_persistent	"1"	PHP_INI_SYSTEM
ifx.max_persistent	"-1"	PHP_INI_SYSTEM
ifx.max_links	"-1"	PHP_INI_SYSTEM
ifx.default_host	NULL	PHP_INI_SYSTEM
ifx.default_user	NULL	PHP_INI_SYSTEM
ifx.default_password	NULL	PHP_INI_SYSTEM
ifx.blobinfile	"1"	PHP_INI_ALL
ifx.textasvarchar	"0"	PHP_INI_ALL
ifx.byteasvarchar	"0"	PHP_INI_ALL
ifx.charasvarchar	"0"	PHP_INI_ALL
ifx.nullformat	"0"	PHP_INI_ALL
ingres.allow_persistent	"1"	PHP_INI_SYSTEM
ingres.max_persistent	"-1"	PHP_INI_SYSTEM
ingres.max_links	"-1"	PHP_INI_SYSTEM
ingres.default_database	NULL	PHP_INI_ALL
ingres.default_user	NULL	PHP_INI_ALL
ingres.default_password	NULL	PHP_INI_ALL
ibase.allow_persistent	"1"	PHP_INI_SYSTEM
ibase.max_persistent	"-1"	PHP_INI_SYSTEM
ibase.max_links	"-1"	PHP_INI_SYSTEM
ibase.default_user	NULL	PHP_INI_ALL
ibase.default_password	NULL	PHP_INI_ALL
ibase.timestampformat	"%m/%d/%Y%H:%M:%S"	PHP_INI_ALL
ibase.dateformat	"%m/%d/%Y"	PHP_INI_ALL
ibase.timeformat	"%H:%M:%S"	PHP_INI_ALL
java.class.path	NULL	PHP_INI_ALL
java.home	NULL	PHP_INI_ALL
java.library.path	NULL	PHP_INI_ALL
java.library	JAVALIB	PHP_INI_ALL

Name	Default	Changeable
java.library	NULL	PHP_INI_ALL
ldap.max_links	"-1"	PHP_INI_SYSTEM
mbstring.detect_order	NULL	PHP_INI_ALL
mbstring.http_input	NULL	PHP_INI_ALL
mbstring.http_output	NULL	PHP_INI_ALL
mbstring.internal_encoding	NULL	PHP_INI_ALL
mbstring.substitute_character	NULL	PHP_INI_ALL
mbstring.func_overload	"0"	PHP_INI_SYSTEM
mcrypt.algorithms_dir	NULL	PHP_INI_ALL
mcrypt.modes_dir	NULL	PHP_INI_ALL
mime_magic.magicfile	"/usr/share/misc/magic.mime"	PHP_INI_SYSTEM
mssql.allow_persistent	"1"	PHP_INI_SYSTEM
mssql.max_persistent	"-1"	PHP_INI_SYSTEM
mssql.max_links	"-1"	PHP_INI_SYSTEM
mssql.min_error_severity	"10"	PHP_INI_ALL
mssql.min_message_severity	"10"	PHP_INI_ALL
mssql.compatability_mode	"0"	PHP_INI_ALL
mssql.connect_timeout	"5"	PHP_INI_ALL
mssql.timeout	"60"	PHP_INI_ALL
mssql.textsize	"-1"	PHP_INI_ALL
mssql.textlimit	"-1"	PHP_INI_ALL
mssql.batchsize	"0"	PHP_INI_ALL
mssql.datetimeconvert	"1"	PHP_INI_ALL
mysql.allow_persistent	"1"	PHP_INI_SYSTEM
mysql.max_persistent	"-1"	PHP_INI_SYSTEM
mysql.max_links	"-1"	PHP_INI_SYSTEM
mysql.default_host	NULL	PHP_INI_ALL
mysql.default_user	NULL	PHP_INI_ALL
mysql.default_password	NULL	PHP_INI_ALL
mysql.default_port	NULL	PHP_INI_ALL
mysql.default_socket	NULL	PHP_INI_ALL
ncurses.value	"42"	PHP_INI_ALL
ncurses.string	"foobar"	PHP_INI_ALL
odbc.allow_persistent	"1"	PHP_INI_SYSTEM
odbc.max_persistent	"-1"	PHP_INI_SYSTEM
odbc.max_links	"-1"	PHP_INI_SYSTEM
odbc.default_db	NULL	PHP_INI_ALL
odbc.default_user	NULL	PHP_INI_ALL
odbc.default_pw	NULL	PHP_INI_ALL
odbc.defaultlrl	"4096"	PHP_INI_ALL
odbc.defaultbinmode	"1"	PHP_INI_ALL
odbc.check_persistent	"1"	PHP_INI_SYSTEM

Name	**Default**	**Changeable**
pfpro.defaulthost	"test.signio.com"	
pfpro.defaulthost	"test-payflow.verisign.com"	
pfpro.defaultport	"443"	PHP_INI_ALL
pfpro.defaulttimeout	"30"	PHP_INI_ALL
pfpro.proxyaddress	""	PHP_INI_ALL
pfpro.proxyport	""	PHP_INI_ALL
pfpro.proxylogon	""	PHP_INI_ALL
pfpro.proxypassword	""	PHP_INI_ALL
pgsql.allow_persistent	"1"	PHP_INI_SYSTEM
pgsql.max_persistent	"-1"	PHP_INI_SYSTEM
pgsql.max_links	"-1"	PHP_INI_SYSTEM
pgsql.auto_reset_persistent	"0"	PHP_INI_SYSTEM
pgsql.ignore_notice	"0"	PHP_INI_ALL
pgsql.log_notice	"0"	PHP_INI_ALL
session.save_path	"/tmp"	PHP_INI_ALL
session.name	"PHPSESSID"	PHP_INI_ALL
session.save_handler	"files"	PHP_INI_ALL
session.auto_start	"0"	PHP_INI_ALL
session.gc_probability	"1"	PHP_INI_ALL
session.gc_maxlifetime	"1440"	PHP_INI_ALL
session.serialize_handler	"php"	PHP_INI_ALL
session.cookie_lifetime	"0"	PHP_INI_ALL
session.cookie_path	"/"	PHP_INI_ALL
session.cookie_domain	""	PHP_INI_ALL
session.cookie_secure	""	PHP_INI_ALL
session.use_cookies	"1"	PHP_INI_ALL
session.referer_check	""	PHP_INI_ALL
session.entropy_file	""	PHP_INI_ALL
session.entropy_length	"0"	PHP_INI_ALL
session.cache_limiter	"nocache"	PHP_INI_ALL
session.cache_expire	"180"	PHP_INI_ALL
session.use_trans_sid	"1"	PHP_INI_ALL
session.encode_sources	"globals	track"
extname.global_value	"42"	PHP_INI_ALL
extname.global_string	"foobar"	PHP_INI_ALL
assert.active	"1"	PHP_INI_ALL
assert.bail	"0"	PHP_INI_ALL
assert.warning	"1"	PHP_INI_ALL
assert.callback	NULL	PHP_INI_ALL
assert.quiet_eval	"0"	PHP_INI_ALL
safe_mode_protected_env_vars	SAFE_MODE_PROTECTED_ENV_VARS	PHP_INI_SYSTEM

Name	Default	Changeable
safe_mode_allowed_env_vars	SAFE_MODE_ALLOWED_ENV_VARS	PHP_INI_SYSTEM
url_rewriter.tags	"a=href	area=href
url_rewriter.tags	"a=href	area=href
sybct.allow_persistent	"1"	PHP_INI_SYSTEM
sybct.max_persistent	"-1"	PHP_INI_SYSTEM
sybct.max_links	"-1"	PHP_INI_SYSTEM
sybct.min_server_severity	"10"	PHP_INI_ALL
sybct.min_client_severity	"10"	PHP_INI_ALL
sybct.hostname	NULL	PHP_INI_ALL
tokenizer.global_value	"42"	PHP_INI_ALL
tokenizer.global_string	"foobar"	PHP_INI_ALL
vpopmail.directory	""	PHP_INI_ALL
zlib.output_compression	"0"	PHP_INI_ALL
zlib.output_compression_level	"-1"	PHP_INI_ALL
define_syslog_variables	"0"	PHP_INI_ALL
highlight.bg	HL_BG_COLOR	PHP_INI_ALL
highlight.comment	HL_COMMENT_COLOR	PHP_INI_ALL
highlight.default	HL_DEFAULT_COLOR	PHP_INI_ALL
highlight.html	HL_HTML_COLOR	PHP_INI_ALL
highlight.keyword	HL_KEYWORD_COLOR	PHP_INI_ALL
highlight.string	HL_STRING_COLOR	PHP_INI_ALL
allow_call_time_pass_reference	"1"	PHP_INI_SYSTEM|PHP_INI_PERDIR
asp_tags	"0"	PHP_INI_SYSTEM|PHP_INI_PERDIR
display_errors	"1"	PHP_INI_ALL
display_startup_errors	"0"	PHP_INI_ALL
enable_dl	"1"	PHP_INI_SYSTEM
expose_php	"1"	PHP_INI_SYSTEM
html_errors	"1"	PHP_INI_SYSTEM
xmlrpc_errors	"0"	PHP_INI_SYSTEM
xmlrpc_error_number	"0"	PHP_INI_ALL
ignore_user_abort	"0"	PHP_INI_ALL
implicit_flush	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
log_errors	"0"	PHP_INI_ALL
log_errors_max_len	"1024"	PHP_INI_ALL
ignore_repeated_errors	"0"	PHP_INI_ALL
ignore_repeated_source	"0"	PHP_INI_ALL
magic_quotes_gpc	"1"	PHP_INI_ALL
magic_quotes_runtime	"0"	PHP_INI_ALL

Name	Default	Changeable
magic_quotes_sybase	"0"	PHP_INI_ALL
output_buffering	"0"	PHP_INI_PERDIR|PHP_INI_SYSTEM
output_handler	NULL	PHP_INI_PERDIR|PHP_INI_SYSTEM
register_argc_argv	"1"	PHP_INI_ALL
register_globals	"0"	PHP_INI_ALL
safe_mode	"1"	PHP_INI_SYSTEM
safe_mode	"0"	PHP_INI_SYSTEM
safe_mode_include_dir	NULL	PHP_INI_SYSTEM
safe_mode_gid	"0"	PHP_INI_SYSTEM
short_open_tag	DEFAULT_SHORT_OPEN_TAG	PHP_INI_SYSTEM|PHP_INI_PERDIR
sql.safe_mode	"0"	PHP_INI_SYSTEM
track_errors	"0"	PHP_INI_ALL
y2k_compliance	"0"	PHP_INI_ALL
unserialize_callback_func	NULL	PHP_INI_ALL
arg_separator.output	"&"	PHP_INI_ALL
arg_separator.input	"&"	PHP_INI_SYSTEM|PHP_INI_PERDIR
auto_append_file	NULL	PHP_INI_ALL
auto_prepend_file	NULL	PHP_INI_ALL
doc_root	NULL	PHP_INI_SYSTEM
default_charset	SAPI_DEFAULT_CHARSET	PHP_INI_ALL
default_mimetype	SAPI_DEFAULT_MIMETYPE	PHP_INI_ALL
error_log	NULL	PHP_INI_ALL
extension_dir	PHP_EXTENSION_DIR	PHP_INI_SYSTEM
gpc_order	"GPC"	PHP_INI_ALL
include_path	PHP_INCLUDE_PATH	PHP_INI_ALL
max_execution_time	"30"	PHP_INI_ALL
open_basedir	NULL	PHP_INI_SYSTEM
safe_mode_exec_dir	"1"	PHP_INI_SYSTEM
upload_max_filesize	"2M"	PHP_INI_SYSTEM
file_uploads	"1"	PHP_INI_ALL
post_max_size	"8M"	PHP_INI_SYSTEM
upload_tmp_dir	NULL	PHP_INI_SYSTEM
user_dir	NULL	PHP_INI_SYSTEM
variables_order	NULL	PHP_INI_ALL
error_append_string	NULL	PHP_INI_ALL
error_prepend_string	NULL	PHP_INI_ALL
SMTP	"localhost"	PHP_INI_ALL
browscap	NULL	PHP_INI_SYSTEM

Name	Default	Changeable
error_reporting	NULL	PHP_INI_ALL
memory_limit	"8M"	PHP_INI_ALL
precision	"14"	PHP_INI_ALL
sendmail_from	NULL	PHP_INI_ALL
sendmail_path	DEFAULT_SENDMAIL_PATH	PHP_INI_SYSTEM
disable_functions	""	PHP_INI_SYSTEM
allow_url_fopen	"1"	PHP_INI_ALL
always_populate_raw_post_data	"0"	PHP_INI_ALL
xbithack	"0"	PHP_INI_ALL
engine	"1"	PHP_INI_ALL
last_modified	"0"	PHP_INI_ALL
child_terminate	"0"	PHP_INI_ALL
async_send	"0"	PHP_INI_ALL

Tabella 2. Definition of PHP_INI_* constants

Constant	Value	Meaning
PHP_INI_USER	1	Entry can be set in user scripts
PHP_INI_PERDIR	2	Entry can be set in `.htaccess`
PHP_INI_SYSTEM	4	Entry can be set in `php.ini` or `httpd.conf`
PHP_INI_ALL	7	Entry can be set anywhere

See also: ini_alter(), ini_get(), and ini_restore()

php_logo_guid (PHP 4 >= 4.0.0)

Gets the logo guid

string **php_logo_guid** (void) \linebreak

> **Nota:** This functionality was added in PHP 4.0.0.

See also: phpinfo(), phpversion(), and phpcredits().

php_sapi_name (PHP 4)

Returns the type of interface between web server and PHP

string **php_sapi_name** (void) \linebreak

php_sapi_name() returns a lowercase string which describes the type of interface between web server and PHP (Server API, SAPI). In CGI PHP, this string is "cgi", in mod_php for Apache, this string is "apache" and so on.

Esempio 1. php_sapi_name() Example

```
$sapi_type = php_sapi_name();
if ($sapi_type == "cgi")
    print "You are using CGI PHP\n";
else
    print "You are not using CGI PHP\n";
```

php_uname (PHP 4 >= 4.0.2)

Returns information about the operating system PHP was built on

string **php_uname** (void) \linebreak

php_uname() returns a string with a description of the operating system PHP is built on.

Esempio 1. php_uname() Example

```
if (substr(php_uname(), 0, 7) == "Windows") {
   die ("Sorry, this script doesn’t run on Windows.\n");
}
```

phpcredits (PHP 4 >= 4.0.0)

Prints out the credits for PHP

void **phpcredits** ([int flag]) \linebreak

This function prints out the credits listing the PHP developers, modules, etc. It generates the appropriate HTML codes to insert the information in a page. `flag` is optional, and it defaults to `CREDITS_ALL`. To generate a custom credits page, you may want to use the `flag` parameter. For example to print the general credits, you will use somewhere in your code:

```
...
```

```
phpcredits(CREDITS_GENERAL);
...
```

And if you want to print the core developers and the documentation group, in a page of its own, you will use:

```
<?php
phpcredits(CREDITS_GROUP + CREDITS_DOCS + CREDITS_FULLPAGE);
?>
```

And if you feel like embedding all the credits in your page, then code like the one below will do it:

```
<html>
 <head>
  <title>My credits page</title>
 </head>
 <body>
  <?php
 // some code of your own
 phpcredits(CREDITS_ALL);
 // some more code
 ?>
 </body>
</html>
```

Tabella 1. Pre-defined phpcredits() flags

name	description
CREDITS_ALL	All the credits, equivalent to using: CREDITS_DOCS + CREDITS_GENERAL + CREDITS_GROUP + CREDITS_MODULES + CREDITS_FULLPAGE. It generates a complete stand-alone HTML page with the appropriate tags.
CREDITS_DOCS	The credits for the documentation team
CREDITS_FULLPAGE	Usually used in combination with the other flags. Indicates that the a complete stand-alone HTML page needs to be printed including the information indicated by the other flags.
CREDITS_GENERAL	General credits: Language design and concept, PHP 4.0 authors and SAPI module.

name	description
CREDITS_GROUP	A list of the core developers
CREDITS_MODULES	A list of the extension modules for PHP, and their authors
CREDITS_SAPI	A list of the server API modules for PHP, and their authors

See also: phpinfo(), phpversion(), and php_logo_guid().

phpinfo (PHP 3, PHP 4 >= 4.0.0)

Outputs lots of PHP information

int **phpinfo** ([int what]) \linebreak

Outputs a large amount of information about the current state of PHP. This includes information about PHP compilation options and extensions, the PHP version, server information and environment (if compiled as a module), the PHP environment, OS version information, paths, master and local values of configuration options, HTTP headers, and the PHP License.

Because every system is setup differently, **phpinfo()** is commonly used to check configuration settings and for available predefined variables on a given system. Also, **phpinfo()** is a valuable debugging tool as it contains all EGPCS (Environment, GET, POST, Cookie, Server) data.

The output may be customized by passing one or more of the following *constants* bitwise values summed together in the optional `what` parameter. One can also combine the respective constants or bitwise values together with the or operator.

Tabella 1. phpinfo() options

Name (constant)	Value	Description
INFO_GENERAL	1	The configuration line, `php.ini` location, build date, Web Server, System and more.
INFO_CREDITS	2	PHP 4 Credits. See also phpcredits().
INFO_CONFIGURATION	4	Current Local and Master values for php directives. See also ini_get().
INFO_MODULES	8	Loaded modules and their respective settings.
INFO_ENVIRONMENT	16	Environment Variable information that's also available in `$_ENV`.
INFO_VARIABLES	32	Shows all predefined variables from EGPCS (Environment, GET, POST, Cookie, Server).

Name (constant)	Value	Description
INFO_LICENSE	64	PHP License information. See also the license faq (http://www.php.net/license/).
INFO_ALL	-1	Shows all of the above. This is the default value.

Esempio 1. phpinfo() examples

```
<?php

// Show all information, defaults to INFO_ALL
phpinfo();

// Show just the module information.
// phpinfo(8) yields identical results.
phpinfo(INFO_MODULES);

?>
```

> **Nota:** Parts of the information displayed are disabled when the `expose_php` configuration setting is set to `off`. This includes the PHP and Zend logos, and the credits.

See also: phpversion(), phpcredits(), php_logo_guid(), ini_get(), ini_set(), and the section on Predefined Variables.

phpversion (PHP 3, PHP 4 >= 4.0.0)

Gets the current PHP version

string **phpversion** (void) \linebreak

Returns a string containing the version of the currently running PHP parser.

> **Nota:** This information is also available in the predefined constant `PHP_VERSION`.

Esempio 1. phpversion() Example

```
<?php
// prints e.g. 'Current PHP version: 4.1.1'
```

```
echo 'Current PHP version: ' . phpversion();
?>
```

See also version_compare(), phpinfo(), phpcredits(), php_logo_guid(), and zend_version().

putenv (PHP 3, PHP 4 >= 4.0.0)

Sets the value of an environment variable

void **putenv** (string setting) \linebreak

Adds *setting* to the server environment. The environment variable will only exist for the duration of the current request. At the end of the request the environment is restored to its original state.

Setting certain environment variables may be a potential security breach. The `safe_mode_allowed_env_vars` directive contains a comma-delimited list of prefixes. In Safe Mode, the user may only alter environment variables whose names begin with the prefixes supplied by this directive. By default, users will only be able to set environment variables that begin with `PHP_` (e.g. `PHP_FOO=BAR`). Note: if this directive is empty, PHP will let the user modify ANY environment variable!

The `safe_mode_protected_env_vars` directive contains a comma-delimited list of environment variables, that the end user won't be able to change using **putenv()**. These variables will be protected even if `safe_mode_allowed_env_vars` is set to allow to change them.

Attenzione

These directives have only effect when safe-mode itself is enabled!

Esempio 1. Setting an Environment Variable

```
putenv ("UNIQID=$uniqid");
```

See also getenv().

set_magic_quotes_runtime (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Sets the current active configuration setting of magic_quotes_runtime

long **set_magic_quotes_runtime** (int new_setting) \linebreak

Set the current active configuration setting of magic_quotes_runtime (0 for off, 1 for on).

See also: get_magic_quotes_gpc() and get_magic_quotes_runtime().

set_time_limit (PHP 3, PHP 4 >= 4.0.0)

Limits the maximum execution time

void **set_time_limit** (int seconds) \linebreak

Set the number of seconds a script is allowed to run. If this is reached, the script returns a fatal error. The default limit is 30 seconds or, if it exists, the `max_execution_time` value defined in the configuration file. If seconds is set to zero, no time limit is imposed.

When called, **set_time_limit()** restarts the timeout counter from zero. In other words, if the timeout is the default 30 seconds, and 25 seconds into script execution a call such as set_time_limit(20) is made, the script will run for a total of 45 seconds before timing out.

set_time_limit() has no effect when PHP is running in safe mode. There is no workaround other than turning off safe mode or changing the time limit in the configuration file.

> **Nota:** The **set_time_limit()** function and the configuration directive max_execution_time only affect the execution time of the script itself. Any time spent on activity that happens outside the execution of the script such as system calls using system(), the sleep() function, database queries, etc. is not included when determining the maximum time that the script has been running.

version_compare (PHP 4 >= 4.1.0)

Compares two "PHP-standardized" version number strings

int **version_compare** (string version1, string version2 [, string operator]) \linebreak

version_compare() compares two "PHP-standardized" version number strings. This is useful if you would like to write programs working only on some versions of PHP.

version_compare() returns -1 if the first version is lower than the second, 0 if they are equal, and +1 if the second is lower.

If you specify the third optional *`operator`* argument, you can test for a particular relationship. The possible operators are: <, `lt`, <=, `le`, >, `gt`, >=, `ge`, ==, =, `eq`, `!=`, <>, `ne` respectively. Using this argument, the function will return 1 if the relationship is the one specified by the operator, 0 otherwise.

Esempio 1. version_compare() Example

```
// prints -1
echo version_compare("4.0.4", "4.0.6");

// these all print 1
echo version_compare("4.0.4", "4.0.6", "<");
```

```
echo version_compare("4.0.6", "4.0.6", "eq");
```

zend_logo_guid (PHP 4 >= 4.0.0)

Gets the zend guid

string **zend_logo_guid** (void) \linebreak

Nota: This functionality was added in PHP 4.0.0.

zend_version (PHP 4 >= 4.0.0)

Gets the version of the current Zend engine

string **zend_version** (void) \linebreak

Returns a string containing the version of the currently running PHP parser.

Esempio 1. zend_version() Example

```
// prints e.g. 'Zend engine version: 1.0.4'
echo "Zend engine version: " . zend_version();
```

See also phpinfo(), phpcredits(), php_logo_guid(), and phpversion().

LXXVIII. POSIX functions

This module contains an interface to those functions defined in the IEEE 1003.1 (POSIX.1) standards document which are not accessible through other means. POSIX.1 for example defined the open(), read(), write() and close() functions, too, which traditionally have been part of PHP 3 for a long time. Some more system specific functions have not been available before, though, and this module tries to remedy this by providing easy access to these functions.

Attenzione

Sensitive data can be retrieved with the POSIX functions, e.g. posix_getpwnam() and friends. None of the POSIX function perform any kind of access checking when safe mode is enabled. It's therefore **strongly** advised to disable the POSIX extension at all (use `--disable-posix` in your configure line) if you're operating in such an environment.

Nota: The POSIX extension is **not** available on the Windows platform.

posix_ctermid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Get path name of controlling terminal

string **posix_ctermid** (void) \linebreak

Needs to be written.

posix_getcwd (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Pathname of current directory

string **posix_getcwd** (void) \linebreak

Needs to be written ASAP.

posix_getegid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return the effective group ID of the current process

int **posix_getegid** (void) \linebreak

Return the numeric effective group ID of the current process. See also posix_getgrgid() for information on how to convert this into a useable group name.

posix_geteuid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return the effective user ID of the current process

int **posix_geteuid** (void) \linebreak

Return the numeric effective user ID of the current process. See also posix_getpwuid() for information on how to convert this into a useable username.

posix_getgid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return the real group ID of the current process

int **posix_getgid** (void) \linebreak

Return the numeric real group ID of the current process. See also posix_getgrgid() for information on how to convert this into a useable group name.

posix_getgrgid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Return info about a group by group id

array **posix_getgrgid** (int gid) \linebreak

Needs to be written.

posix_getgrnam (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Return info about a group by name

array **posix_getgrnam** (string name) \linebreak

Needs to be written.

posix_getgroups (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return the group set of the current process

array **posix_getgroups** (void) \linebreak

Returns an array of integers containing the numeric group ids of the group set of the current process. See also posix_getgrgid() for information on how to convert this into useable group names.

posix_getlogin (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Return login name

string **posix_getlogin** (void) \linebreak

Returns the login name of the user owning the current process. See posix_getpwnam() for information how to get more information about this user.

posix_getpgid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Get process group id for job control

int **posix_getpgid** (int pid) \linebreak

Returns the process group identifier of the process *pid*.

This is not a POSIX function, but is common on BSD and System V systems. If your system does not support this function at system level, this PHP function will always return FALSE.

posix_getpgrp (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return the current process group identifier

int **posix_getpgrp** (void) \linebreak

Return the process group identifier of the current process. See POSIX.1 and the getpgrp(2) manual page on your POSIX system for more information on process groups.

posix_getpid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return the current process identifier

int **posix_getpid** (void) \linebreak

Return the process identifier of the current process.

posix_getppid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return the parent process identifier

int **posix_getppid** (void) \linebreak

Return the process identifier of the parent process of the current process.

posix_getpwnam (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Return info about a user by username

array **posix_getpwnam** (string username) \linebreak

Returns an associative array containing information about a user referenced by an alphanumeric username, passed in the *username* parameter.

The array elements returned are:

Tabella 1. The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not her real, full name. This should be the same as the *username* parameter used when calling the function, and hence redundant.

Element	Description
passwd	The passwd element contains the user’s password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID of the user in numeric form.
gid	The group ID of the user. Use the function posix_getgrgid() to resolve the group name and a list of its members.
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing the user’s full name, office phone, office number, and home phone number. On most systems, only the user’s full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user’s default shell.

posix_getpwuid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Return info about a user by user id

array **posix_getpwuid** (int uid) \linebreak

Returns an associative array containing information about a user referenced by a numeric user ID, passed in the *uid* parameter.

The array elements returned are:

Tabella 1. The user information array

Element	Description
name	The name element contains the username of the user. This is a short, usually less than 16 character "handle" of the user, not her real, full name.
passwd	The passwd element contains the user’s password in an encrypted format. Often, for example on a system employing "shadow" passwords, an asterisk is returned instead.
uid	User ID, should be the same as the *uid* parameter used when calling the function, and hence redundant.

Element	Description
gid	The group ID of the user. Use the function posix_getgrgid() to resolve the group name and a list of its members.
gecos	GECOS is an obsolete term that refers to the finger information field on a Honeywell batch processing system. The field, however, lives on, and its contents have been formalized by POSIX. The field contains a comma separated list containing the user's full name, office phone, office number, and home phone number. On most systems, only the user's full name is available.
dir	This element contains the absolute path to the home directory of the user.
shell	The shell element contains the absolute path to the executable of the user's default shell.

posix_getrlimit (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return info about system ressource limits

array **posix_getrlimit** (void) \linebreak

Needs to be written ASAP.

posix_getsid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Get the current sid of the process

int **posix_getsid** (int pid) \linebreak

Return the sid of the process *pid*. If *pid* is 0, the sid of the current process is returned.

This is not a POSIX function, but is common on System V systems. If your system does not support this function at system level, this PHP function will always return FALSE.

posix_getuid (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Return the real user ID of the current process

int **posix_getuid** (void) \linebreak

Return the numeric real user ID of the current process. See also posix_getpwuid() for information on how to convert this into a useable username.

posix_isatty (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Determine if a file descriptor is an interactive terminal

bool **posix_isatty** (int fd) \linebreak

Needs to be written.

posix_kill (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Send a signal to a process

bool **posix_kill** (int pid, int sig) \linebreak

Send the signal *sig* to the process with the process identifier *pid*. Returns FALSE, if unable to send the signal, TRUE otherwise.

See also the kill(2) manual page of your POSIX system, which contains additional information about negative process identifiers, the special pid 0, the special pid -1, and the signal number 0.

posix_mkfifo (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Create a fifo special file (a named pipe)

bool **posix_mkfifo** (string pathname, int mode) \linebreak

posix_mkfifo() creates a special FIFO file which exists in the file system and acts as a bidirectional communication endpoint for processes.

The second parameter *mode* has to be given in octal notation (e.g. 0644). The permission of the newly created FIFO also depends on the setting of the current umask(). The permissions of the created file are (mode & ~umask).

> **Nota:** Quando safe-mode è abilitato, PHP controlla che la directory nella quale si sta lavorando, abbia lo stesso UID dello script che è in esecuzione.

posix_setegid (PHP 4 >= 4.0.2)

Set the effective GID of the current process

bool **posix_setegid** (int gid) \linebreak

Set the effective group ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns TRUE on success, FALSE otherwise.

posix_seteuid (PHP 4 >= 4.0.2)

Set the effective UID of the current process

bool **posix_seteuid** (int uid) \linebreak

Set the real user ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns `TRUE` on success, `FALSE` otherwise. See also posix_setgid().

posix_setgid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Set the GID of the current process

bool **posix_setgid** (int gid) \linebreak

Set the real group ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function. The appropriate order of function calls is **posix_setgid()** first, posix_setuid() last.

Returns `TRUE` on success, `FALSE` otherwise.

posix_setpgid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

set process group id for job control

int **posix_setpgid** (int pid, int pgid) \linebreak

Let the process *`pid`* join the process group *`pgid`*. See POSIX.1 and the setsid(2) manual page on your POSIX system for more informations on process groups and job control. Returns `TRUE` on success, `FALSE` otherwise.

posix_setsid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Make the current process a session leader

int **posix_setsid** (void) \linebreak

Make the current process a session leader. See POSIX.1 and the setsid(2) manual page on your POSIX system for more informations on process groups and job control. Returns the session id.

posix_setuid (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Set the UID of the current process

bool **posix_setuid** (int uid) \linebreak

Set the real user ID of the current process. This is a privileged function and you need appropriate privileges (usually root) on your system to be able to perform this function.

Returns TRUE on success, FALSE otherwise. See also posix_setgid().

posix_times (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Get process times

array **posix_times** (void) \linebreak

Returns a hash of strings with information about the current process CPU usage. The indices of the hash are

- ticks - the number of clock ticks that have elapsed since reboot.
- utime - user time used by the current process.
- stime - system time used by the current process.
- cutime - user time used by current process and children.
- cstime - system time used by current process and children.

posix_ttyname (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Determine terminal device name

string **posix_ttyname** (int fd) \linebreak

Needs to be written.

posix_uname (PHP 3>= 3.0.10, PHP 4 >= 4.0.0)

Get system name

array **posix_uname** (void) \linebreak

Returns a hash of strings with information about the system. The indices of the hash are

- sysname - operating system name (e.g. Linux)
- nodename - system name (e.g. valiant)
- release - operating system release (e.g. 2.2.10)
- version - operating system version (e.g. #4 Tue Jul 20 17:01:36 MEST 1999)
- machine - system architecture (e.g. i586)
- domainname - DNS domainname (e.g. php.net)

domainname is a GNU extension and not part of POSIX.1, so this field is only available on GNU systems or when using the GNU libc.

Posix requires that you must not make any assumptions about the format of the values, e.g. you cannot rely on three digit version numbers or anything else returned by this function.

LXXIX. Funzioni PostgreSQL

Postgres, sviluppato originariamente nel Dipartimento di Informatica dell'Unniversità di Berkeley, ha anticipato molti dei concetti su oggetti e relazioni che ora stanno diventando disponibili in alcuni database commerciali. Postgres fornisce supporto per il linguaggio SQL/92/SQL99, integrità transazionale ed estensibilità dei tipi. PostgreSQL è un discendente open source dell'originario codice di Berkeley.

Il database PostgreSQL è un prodotto Open Source ed è disponibile gratuitamente. Per utilizzare il supporto a PostgreSQL, occorre PosgreSQL 6.5 o versioni più recenti. PostgreSQL 7.0 o successivi permettono di abilitare tutte le possibilità di questo modulo. PostgreSQL ammette molte codifiche di carattere, tra cui la codifica multibyte. La versione corrente e maggiori informazioni su PostgreSQL sono disponibili su http://www.postgresql.org/.

Per abilitare il supporto PostgreSQL, `--with-pgsql[=DIR]` è richiesto quando si compila il PHP. Se è disponibile il modulo di libreria dinamico, questo può essere caricato usando la direttiva extension in `php.ini` o la funzione dl(). Le direttive ini previste sono descritte in `php.ini-dist`

che è disribuito con il pacchetto dei sorgenti.

Attenzione

L'utilizzo del modulo PostgreSQL con PHP 4.0.6 non è raccomandato a causa di un bug nella gestione dei messaggi. Si usi PHP 4.1.0 o successivi.

Attenzione

I nomi delle funzioni relative a PostgreSQL verranno cambiate a partire dalla versione 4.2.0 per conformarsi agli standard di sviluppo attuali. La maggior parte dei nuovi nomi avrà underscore aggiuntivi, per esempio pg_lo_open(). Alcune funzioni verranno rinominate per dare consistenza. Per esempio pg_exec() diventerà pg_query(). I vecchi nomi potranno essere usati nella versione 4.2.0 e in alcune versioni successive alla 4.2.0, ma potranno essere cancellate in futuro.

Tabella 1. Nomi di funzione cambiati

Vecchio nome	Nuovo nome
pg_exec()	pg_query()
pg_getlastoid()	pg_last_oid()
pg_cmdtuples()	pg_affected_rows()
pg_numrows()	pg_num_rows()
pg_numfields()	pg_num_fields()
pg_fieldname()	pg_field_name()
pg_fieldsize()	pg_field_size()
pg_fieldnum()	pg_field_num()
pg_fieldprtlen()	pg_field_prtlen()
pg_fieldisnull()	pg_field_is_null()
pg_freeresult()	pg_free_result()
pg_result()	pg_fetch_result()
pg_loreadall()	pg_lo_read_all()
pg_locreate()	pg_lo_create()
pg_lounlink()	pg_lo_unlink()
pg_loopen()	pg_lo_open()
pg_loclose()	pg_lo_close()
pg_loread()	pg_lo_read()
pg_lowrite()	pg_lo_write()
pg_loimport()	pg_lo_import()
pg_loexport()	pg_lo_export()

La vecchia sintassi di pg_connect()/pg_pconnect() sarà deprecata per supportare, in futuro, connessioni asincrone. Si usi una stringa di connessione con pg_connect() e pg_pconnect().

Non tutte le funzioni sono supportate su tutte le architetture. Dipende dalla versione di libpq

(L’interfaccia Client C per PostgreSQL) e da come libpq è compilato. Se c’è una funzione mancante, libpq non supporta la feature richiesta per quella funzione.

È importante usare versioni di libpq più recenti rispetto al Server PostgreSQL al quale ci si vuole collegare. Se si usa una versione di libpq più vecchia rispetto al Server PostgreSQL al quale ci si vuole collegare, si andrà probabilmente incontro a problemi.

Fin dalla versione 6.3 (03/02/1998) PostgreSQL usa gli unix domain socket di default. La porta TCP, di default, NON verrà aperta. La tabella sottostante descrive queste nuove possibilità di connessione. Questo socket può essere trovato in `/tmp/.s.PGSQL.5432`. Questa opzione può venire abilitata con la flag ’-i’ a **postmaster** e il suo significato è: "ascolta i socket TCP/IP così come gli Unix domain socket".

Tabella 2. Postmaster e PHP

Postmaster	PHP	Status
postmaster &	pg_connect("dbname=NomeMioDatabase");	OK
postmaster -i &	pg_connect("dbname=NomeMioDatabase");	OK
postmaster &	pg_connect("host=localhost dbname=NomeMioDatabase");	Unable to connect to PostgreSQL server: connectDB() failed: Is the postmaster running and accepting TCP/IP (with -i) connection at ’localhost’ on port ’5432’? in /path/to/file.php on line 20.
postmaster -i &	pg_connect("host=localhost dbname=NomeMioDatabase");	OK

Si può stabilire una connessione al server PostgreSQL usando le seguenti coppie di valori impostati nella stringa di comando: **$conn = pg_connect("host=IlMioHost port=LaMiaPort tty=myTTY options=LeMieOpzioni dbname=IlMioDB user=IlMioUtente password=LaMiaPassword ");**

L’uso della sintassi in uso precedentemente: **$conn = pg_connect ("host", "port", "options", "tty", "dbname")** è deprecato.

Le variabili d’ambiente modificano il comporetamento server/client di PostgreSQL. Per esempio, il modulo PostgreSQL si baserà sulla variabile PGHOST quando il nome dell’host è omesso nella stringa di connessione. Le variabili d’ambiente riconosciute sono diverse da versione a versione. Consultare il Manuale del Programmatore PostgreSQL (lbpq - Variabili d’Ambiente) per ulteriori dettagli.

Assicurarsi di aver impostato le variabili d’ambiente per l’utente appropriato. Usare `$_ENV` o getenv() per controllare quali variabili d’ambiente sono disponibili al processo corrente.

Esempio 1. Impostazione dei parametri di default

```
PGHOST=pgsql.example.com
PGPORT=7890
PGDATABASE=web-system
PGUSER=web-user
```

```
PGPASSWD=segreta
PGDATESTYLE=ISO
PGTZ=JST
PGCLIENTENCODING=EUC-JP

export PGHOST PGPORT PGDATABASE PGUSER PGPASSWORD PGDATESTYLE PGTZ PGCLIENTENCODING
```

A partire da PostgreSQL 7.1.0, il tipo di dato text ha 1GB come dimensione massima. Nelle versioni più vecchie di PostgreSQL il tipo di dato text è limitato dalla dimensione del block. (Default 8KB. Massimo 32KB, specificato al momento della compilazione)

Per usare l'interfaccia large object (lo), è necessario includerla entro un blocco di una transazione. Un blocco di transazione inizia con un comando SQL **begin** e se la transazione è stata valida, termina con **commit** o **end**. Se la transazione fallisce, essa deve venire chiusa con **rollback** o **abort**.

Esempio 2. Utilizzare Large Objects

```
<?php
    $database = pg_connect ("dbname=jacarta");
    pg_query ($database, "begin");
    $oid = pg_lo_create ($database);
    echo "$oid\n";
    $handle = pg_lo_open ($database, $oid, "w");
    echo "$handle\n";
    pg_lo_write ($handle, "dati large object");
    pg_lo_close ($handle);
    pg_query ($database, "commit");
?>
```

Non chiudere la risorsa di connessione prima di aver chiuso la risorsa large object.

pg_affected_rows (PHP 4 >= 4.2.0)

Returns number of affected records(tuples)

int **pg_affected_rows** (resource result) \linebreak

pg_affected_rows() returns the number of tuples (instances/records/rows) affected by INSERT, UPDATE, and DELETE queries executed by pg_query(). If no tuple is affected by this function, it will return 0.

Esempio 1. pg_affected_rows()

```
<?php
     $result = pg_query ($conn, "INSERT INTO publisher VALUES ('Author')");
     $cmdtuples = pg_affected_rows ($result);
     echo $cmdtuples . " tuples are affected.";
?>
```

Nota: This function used to be called `pg_cmdtuples()`.

See also pg_query() and pg_num_rows().

pg_cancel_query (PHP 4 >= 4.2.0)

Cancel async query

bool **pg_cancel_query** (resource connection) \linebreak

pg_cancel_query() cancel asynchronous query sent by pg_send_query(). You cannot cancel query executed by pg_query().

See also pg_send_query(), **pg_cancel_result()** and pg_connection_busy()

pg_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Get the client encoding

string **pg_client_encoding** ([resource connection]) \linebreak

pg_client_encoding() returns the client encoding as the string. The returned string should be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250.

Nota: This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. If libpq is compiled without multibyte encoding support, pg_set_client_encoding() always return

"SQL_ASCII". Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details to enable multibyte support and encoding supported.

The function used to be called **pg_clientencoding()**.

See also pg_set_client_encoding().

pg_close (PHP 3, PHP 4 >= 4.0.0)

Chiude una connessione PostgreSQL

bool **pg_close** (resource connection) \linebreak

pg_close() closes down the non-persistent connection to a PostgreSQL database associated with the given *connection* resource. It returns TRUE, if *connection* is a valid connection resource, otherwise it return FALSE.

> **Nota: pg_close()** is not usually necessary, as non-persistent open links are automatically closed at the end of the script's execution. **pg_close()** will not close persistent links generated by pg_pconnect().

If there is open large object resource on the connection, do not close the connection before closing all large object resources.

pg_connect (PHP 3, PHP 4 >= 4.0.0)

Open a PostgreSQL connection

resource **pg_connect** (string connection_string) \linebreak

pg_connect() returns a connection resource that is needed by other PostgreSQL functions.

pg_connect() opens a connection to a PostgreSQL database specified by *connection_string*. It returns a connection resource on success. It returns FALSE, if the connection could not be made. *connection_string* should be a quoted string.

Esempio 1. Using pg_connect

```
<?php
$dbconn = pg_connect ("dbname=mary");
//connect to a database named "mary"
$dbconn2 = pg_connect ("host=localhost port=5432 dbname=mary");
// connect to a database named "mary" on "localhost" at port "5432"
$dbconn3 = pg_connect ("host=sheep port=5432 dbname=mary user=lamb password=foo");
//connect to a database named "mary" on the host "sheep" with a username and password
$conn_string = "host=sheep port=5432 dbname=test user=lamb password=bar";
$dbconn4 = pg_connect ($conn_string);
//connect to a database named "test" on the host "sheep" with a username and password
```

```
?>
```

The arguments available for *`connection_string`* includes *`host`*, *`port`*, *`tty`*, *`options`*, *`dbname`*, *`user`*, and *`password`*.

If a second call is made to **pg_connect()** with the same *`connection_string`* arguments, no new connection will be established, but instead, the connection resource of the already opened connection will be returned. You can have multiple connections to the same database if you use different connection string.

Syntax supports multiple parameters: **$conn = pg_connect ("host", "port", "options", "tty", "dbname")** has been deprecated.

See also pg_pconnect(), pg_close(), pg_host(), pg_port(), pg_tty(), pg_options() and pg_dbname().

pg_connection_busy (PHP 4 >= 4.2.0)

Get connection is busy or not

bool **pg_connection_busy** (resource connection) \linebreak

pg_connection_busy() returns `TRUE` if connection busy. If connection is busy, previously sent query to PostgreSQL server is still executing. If pg_get_result() is called, pg_get_result() will block.

See also pg_connection_status() and pg_get_result()

pg_connection_reset (PHP 4 >= 4.2.0)

Reset connection (reconnect)

bool **pg_connection_reset** (resource connection) \linebreak

pg_connection_reset() reset connection with the same parameter when connection is made. It is useful for error recovery. It returns `TRUE` if it resets connection successfully, otherwise returns `FALSE`.

See also pg_connect(), pg_pconnect() and pg_connection_status()

pg_connection_status (PHP 4 >= 4.2.0)

Get connection status

int **pg_connection_status** (resource connection) \linebreak

pg_connection_status() returns a connection status. Possible status is `PGSQL_CONNECTION_O` or `PGSQL_CONNECTION_BAD`.

See also pg_connection_busy()

pg_convert (PHP 4 CVS only)

Convert associative array value into suitable for SQL statement.

array **pg_convert** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_convert() check and convert `assoc_array` suitable for SQL statement.

> **Nota:** This function is experimental.

See also pg_metadata()

pg_copy_from (PHP 4 >= 4.2.0)

Copy table from array

int **pg_copy_from** (int connection, string table_name, array rows [, string delimiter [, string null_as]]) \linebreak

pg_copy_from() copy table from array. It return `TRUE` for success, otherwise `FALSE`.

See also pg_copy_to()

pg_copy_to (PHP 4 >= 4.2.0)

Copy table to array

int **pg_copy_to** (int connection, string table_name [, string delimiter [, string null_as]]) \linebreak

pg_copy_to() copy table to array. The result array is returned if it success to copy. Otherwise it returns `FALSE`.

See also pg_copy_from()

pg_dbname (PHP 3, PHP 4 >= 4.0.0)

Get the database name

string **pg_dbname** (resource connection) \linebreak

pg_dbname() returns the name of the database that the given PostgreSQL *`connection`* resource. It returns `FALSE`, if *`connection`* is not a valid PostgreSQL connection resource.

pg_delete (PHP 4 CVS only)

Delete records.

long **pg_delete** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_delete() deletes record condition specified by `assoc_array` which has `field=>value`. If `option` is specified, pg_convert() is applied to `assoc_array` with specified option.

Esempio 1. pg_delete

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted autotmatically
    $res = pg_delete($db, 'post_log', $_POST);
    if ($res) {
        echo "POST data is deleted: $res\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Nota: This function is experimental.

See also pg_convert()

pg_end_copy (PHP 4 >= 4.0.3)

Sync with PostgreSQL backend

bool **pg_end_copy** ([resource connection]) \linebreak

pg_end_copy() syncs PostgreSQL frontend (usually a web server process) with the PostgreSQL server after doing a copy operation performed by pg_put_line(). **pg_end_copy()** must be issued, otherwise the PostgreSQL server may get "out of sync" error with the frontend. It returns `TRUE` for success, otherwise it returns `FALSE`.

For further details and an example, see also pg_put_line().

pg_escape_bytea (PHP 4 >= 4.2.0)

Escape binary for bytea type

string **pg_escape_bytea** (string data) \linebreak

pg_escape_string() escapes string for bytea datatype. It returns escaped string.

> **Nota:** This function is requires PostgreSQL 7.2 or later.

See also pg_escape_string()

pg_escape_string (PHP 4 >= 4.2.0)

Escape string for text/char type

string **pg_escape_string** (string data) \linebreak

pg_escape_string() escapes string for text/char datatype. It returns escaped string.

> **Nota:** This function is requires PostgreSQL 7.2 or later.

See also pg_escape_bytea()

pg_fetch_array (PHP 3>= 3.0.1, PHP 4 >= 4.0.0)

Fetch a row as an array

array **pg_fetch_array** (resource result, int row [, int result_type]) \linebreak

pg_fetch_array() returns an array that corresponds to the fetched row (tuples/records). It returns FALSE, if there are no more rows.

pg_fetch_array() is an extended version of pg_fetch_row(). In addition to storing the data in the numeric indices (field index) to the result array, it also stores the data in associative indices (field name) by default.

row is row (record) number to be retrieved. First row is 0.

result_type is optional parameter controls how return value is initialized. *result_type* is a constant and can take the following values: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH. **pg_fetch_array()** returns associative array that has field name as key for PGSQL_ASSOC. field index as key with PGSQL_NUM and both field name/index as key with PGSQL_BOTH. Default is PGSQL_BOTH.

> **Nota:** *result_type* was added in PHP 4.0.

pg_fetch_array() is NOT significantly slower than using pg_fetch_row(), while it provides a significant ease of use.

See also pg_fetch_row() and pg_fetch_object() and **pg_result()**.

Esempio 1. PostgreSQL fetch array

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$arr = pg_fetch_array ($result, 0, PGSQL_NUM);
echo $arr[0] . " <- array\n";

$arr = pg_fetch_array ($result, 1, PGSQL_ASSOC);
echo $arr["author"] . " <- array\n";
?>
```

Nota: From 4.1.0, *row* became optional.

pg_fetch_object (PHP 3>= 3.0.1, PHP 4 >= 4.0.0)

Fetch a row as an object

object **pg_fetch_object** (resource result, int row [, int result_type]) \linebreak

pg_fetch_object() returns an object with properties that correspond to the fetched row. It returns FALSE if there are no more rows or error.

pg_fetch_object() is similar to pg_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

result_type is optional parameter controls how return value is initialized. *result_type* is a constant and can take the following values: PGSQL_ASSOC, PGSQL_NUM, and PGSQL_BOTH. pg_fetch_array() returns associative array that has field name as key for PGSQL_ASSOC. field index as key with PGSQL_NUM and both field name/index as key with PGSQL_BOTH. Default is PGSQL_BOTH.

Nota: *result_type* was added in PHP 4.0.

Speed-wise, the function is identical to pg_fetch_array(), and almost as quick as pg_fetch_row() (the difference is insignificant).

See also **pg_exec()**, pg_fetch_array(), pg_fetch_row() and **pg_result()**.

Esempio 1. Postgres fetch object

```
<?php
$database = "verlag";
$db_conn = pg_connect ("host=localhost port=5432 dbname=$database");
if (!$db_conn): ?>
    <H1>Failed connecting to postgres database <?php echo $database ?></H1> <?php
    exit;
endif;

$qu = pg_exec ($db_conn, "SELECT * FROM verlag ORDER BY autor");
$row = 0; // postgres needs a row counter other dbs might not

while ($data = pg_fetch_object ($qu, $row)) {
    echo $data->autor." (";
    echo $data->jahr ."): ";
    echo $data->titel."<BR>";
    $row++;
}
?>
<PRE>
<?php
$fields[] = Array ("autor", "Author");
$fields[] = Array ("jahr",  "  Year");
$fields[] = Array ("titel", " Title");

$row= 0; // postgres needs a row counter other dbs might not
while ($data = pg_fetch_object ($qu, $row)) {
    echo "----------\n";
    reset ($fields);
    while (list (,$item) = each ($fields)):
        echo $item[1].": ".$data->$item[0]."\n";
    endwhile;
    $row++;
}
echo "----------\n";
?>
</PRE>
<?php
pg_freeresult ($qu);
pg_close ($db_conn);
?>
```

Nota: From 4.1.0, *row* became optional.

pg_fetch_result (PHP 4 >= 4.2.0)

Returns values from a result resource

mixed **pg_fetch_result** (resource result, int row, mixed field) \linebreak

pg_fetch_result() returns values from a *result* resource returned by pg_query(). *row* is integer. *field* is field name(string) or field index (integer). The *row* and *field* specify what cell in the table of results to return. Row numbering starts from 0. Instead of naming the field, you may use the field index as an unquoted number. Field indices start from 0.

PostgreSQL has many built in types and only the basic ones are directly supported here. All forms of integer, boolean and void types are returned as integer values. All forms of float, and real types are returned as float values. All other types, including arrays are returned as strings formatted in the same default PostgreSQL manner that you would see in the **psql** program.

pg_fetch_row (PHP 3>= 3.0.1, PHP 4 >= 4.0.0)

Get a row as an enumerated array

array **pg_fetch_row** (resource result, int row) \linebreak

pg_fetch_row() fetches one row of data from the result associated with the specified *result* resource. The row (record) is returned as an array. Each result column is stored in an array offset, starting at offset 0.

It returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

See also: **pg_exec()**, pg_fetch_array(), pg_fetch_object() and **pg_result()**.

Esempio 1. Postgres fetch row

```
<?php
$conn = pg_pconnect ("dbname=publisher");
if (!$conn) {
    echo "An error occured.\n";
    exit;
}

$result = pg_Exec ($conn, "SELECT * FROM authors");
if (!$result) {
    echo "An error occured.\n";
    exit;
}

$num = pg_numrows($result);

for ($i=0; $i < $num; $i++) {
  $r = pg_fetch_row($result, $i);

  for ($j=0; $j < count($r); $j++) {
    echo "$r[$j] ";
  }
```

```
  echo "<BR>";
}
?>
```

Nota: From 4.1.0, *row* became optional.

pg_field_is_null (PHP 4 >= 4.2.0)

Test if a field is `NULL`

int **pg_field_is_null** (resource result, int row, mixed field) \linebreak

pg_field_is_null() test if a field is `NULL` or not. It returns 1 if the field in the given row is `NULL`. It returns 0 if the field in the given row is NOT `NULL`. Field can be specified as column index (number) or fieldname (string). Row numbering starts at 0.

Nota: This function used to be called `pg_fieldisnull()`.

pg_field_name (PHP 4 >= 4.2.0)

Returns the name of a field

string **pg_field_name** (resource result, int field_number) \linebreak

pg_field_name() returns the name of the field occupying the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts from 0.

Nota: This function used to be called `pg_fieldname()`.

See also pg_field_num().

pg_field_num (PHP 4 >= 4.2.0)

Returns the field number of the named field

int **pg_field_num** (resource result, string field_name) \linebreak

pg_field_num() will return the number of the column (field) slot that corresponds to the *field_name* in the given PostgreSQL *result* resource. Field numbering starts at 0. This function will return -1 on error.

> **Nota:** This function used to be called `pg_fieldnum()`.

See also pg_field_name().

pg_field_prtlen (PHP 4 >= 4.2.0)

Returns the printed length

int **pg_field_prtlen** (resource result, int row_number, string field_name) \linebreak

pg_field_prtlen() returns the actual printed length (number of characters) of a specific value in a PostgreSQL *result*. Row numbering starts at 0. This function will return -1 on an error.

> **Nota:** This function used to be called `pg_field_prtlen()`.

See also pg_field_size().

pg_field_size (PHP 4 >= 4.2.0)

Returns the internal storage size of the named field

int **pg_field_size** (resource result, int field_number) \linebreak

pg_field_size() returns the internal storage size (in bytes) of the field number in the given PostgreSQL *result*. Field numbering starts at 0. A field size of -1 indicates a variable length field. This function will return FALSE on error.

> **Nota:** This function used to be called `pg_fieldsize()`.

See also **pg_field_len()** and pg_field_type().

pg_field_type (PHP 4 >= 4.2.0)

Returns the type name for the corresponding field number

string **pg_field_type** (resource result, int field_number) \linebreak

pg_field_type() returns a string containing the type name of the given *field_number* in the given PostgreSQL *result* resource. Field numbering starts at 0.

Nota: This function used to be called `pg_field_type()`.

See also **pg_field_len()** and pg_field_name().

pg_free_result (PHP 4 >= 4.2.0)

Free result memory

bool **pg_free_result** (resource result) \linebreak

pg_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script is finished. But, if you are sure you are not going to need the result data anymore in a script, you may call **pg_free_result()** with the *`result`* resource as an argument and the associated result memory will be freed. It returns true on success and false if an error occurs.

Nota: This function used to be called `pg_field_len()`.

See also pg_query().

pg_get_result (PHP 4 >= 4.2.0)

Get asynchronous query result

resource **pg_get_result** ([resource connection]) \linebreak

pg_get_result() get result resource from async query executed by pg_send_query(). pg_send_query() can send multiple queries to PostgreSQL server and **pg_get_result()** is used to get query result one by one. It returns result resource. If there is no more results, it returns `FALSE`.

pg_host (PHP 3, PHP 4 >= 4.0.0)

Returns the host name associated with the connection

string **pg_host** (resource connection) \linebreak

pg_host() returns the host name of the given PostgreSQL *`connection`* resource is connected to.

See also pg_connect() and pg_pconnect().

pg_insert (PHP 4 CVS only)

Insert array into table.

bool **pg_insert** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_insert() inserts `assoc_array` which has `field=>value` into table specified as `table_name`. If `options` is specified, pg_convert() is applied to `assoc_array` with specified option.

Esempio 1. pg_insert

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted autotmatically
    $res = pg_insert($db, 'post_log', $_POST);
    if ($res) {
        echo "POST data is succesfully logged\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Nota: This function is experimental.

See also pg_convert()

pg_last_error (PHP 4 >= 4.2.0)

Get the last error message string of a connection

string **pg_last_error** (resource connection) \linebreak

pg_last_error() returns the last error message for given *connection*.

Error messages may be overwritten by internal PostgreSQL(libpq) function calls. It may not return appropriate error message, if multiple errors are occured inside a PostgreSQL module function.

Use pg_result_error(), pg_result_status() and pg_connection_status() for better error handling.

Nota: This function used to be called `pg_errormessage()`.

See also pg_result_error().

pg_last_notice (PHP 4 >= 4.0.6)

Returns the last notice message from PostgreSQL server

string **pg_last_notice** (resource connection) \linebreak

pg_last_notice() returns the last notice message from PostgreSQL server specified by *connection*. PostgreSQL server set notice message when transaction cannot be continued. There one can avoid issuing useless SQL using **pg_exec()** using **pg_last_notice()**. There are other cases that PostgreSQL server sets notice message. Programmer must check contents of notice message if it is related to transaction or not.

Attenzione

This function is EXPERIMENTAL and it is not fully implemented yet. **pg_last_notice()** is added form PHP 4.0.6. However, PHP 4.0.6 has problem with notice message handling. Use of PostgreSQL module with PHP 4.0.6 is not recommended even if you are not using **pg_last_notice()**.

See also **pg_exec()** and **pg_errormessage()**.

pg_last_oid (PHP 4 >= 4.2.0)

Returns the last object's oid

int **pg_last_oid** (resource result) \linebreak

pg_last_oid() is used to retrieve the `oid` assigned to an inserted tuple (record) if the result resource is used from the last command sent via pg_query() and was an SQL INSERT. Returns a positive integer if there was a valid `oid`. It returns `FALSE` if an error occurs or the last command sent via pg_query() was not an INSERT or INSERT is failed.

OID field became an optional field from PostgreSQL 7.2. When OID field is not defined in a table, programmer must use pg_result_status() to check if record is is inserted successfully or not.

Nota: This function used to be called `pg_getlastoid()`.

See also pg_query() and pg_result_status()

pg_lo_close (PHP 4 >= 4.2.0)

Close a large object

bool **pg_lo_close** (resource large_object) \linebreak

pg_lo_close() closes a Large Object. *large_object* is a resource for the large object from pg_lo_open().

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called `pg_loclose()`.

See also pg_lo_open(), pg_lo_create() and pg_lo_import().

pg_lo_create (PHP 4 >= 4.2.0)

Create a large object

int **pg_lo_create** (resource connection) \linebreak

pg_lo_create() creates a Large Object and returns the `oid` of the large object. *`connection`* specifies a valid database connection opened by pg_connect() or pg_pconnect(). PostgreSQL access modes INV_READ, INV_WRITE, and INV_ARCHIVE are not supported, the object is created always with both read and write access. INV_ARCHIVE has been removed from PostgreSQL itself (version 6.3 and above). It returns large object oid otherwise. It returns `FALSE`, if an error occurred,

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called `pg_locreate()`.

pg_lo_export (PHP 4 >= 4.2.0)

Export a large object to file

bool **pg_lo_export** (int oid, string pathname [, resource connection]) \linebreak

The *`oid`* argument specifies oid of the large object to export and the *`pathname`* argument specifies the pathname of the file. It returns `FALSE` if an error occurred, `TRUE` otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: This function used to be called `pg_loexport()`.

See also pg_lo_import().

pg_lo_import (PHP 4 >= 4.2.0)

Import a large object from file

int **pg_lo_import** (string pathname [, resource connection]) \linebreak

The *`pathname`* argument specifies the pathname of the file to be imported as a large object. It returns `FALSE` if an error occurred, oid of the just created large object otherwise.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

Nota: Quando safe-mode è abilitato, PHP controlla che i file o le directory sulle quali si sta andando a lavorare, abbiano lo stesso UID dello script che è in esecuzione.

Nota: This function used to be called `pg_loimport()`.

See also pg_lo_export() and pg_lo_open().

pg_lo_open (PHP 4 >= 4.2.0)

Open a large object

resource **pg_lo_open** (resource connection, int oid, string mode) \linebreak

pg_lo_open() open a Large Object and returns large object resource. The resource encapsulates information about the connection. *oid* specifies a valid large object oid and *mode* can be either "r", "w", or "rw". It returns FALSE if there is an error.

> **Attenzione**
>
> Do not close the database connection before closing the large object resource.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called `pg_loopen()`.

See also pg_lo_close() and pg_lo_create().

pg_lo_read (PHP 4 >= 4.2.0)

Read a large object

string **pg_lo_read** (resource large_object, int len) \linebreak

pg_lo_read() reads at most *len* bytes from a large object and returns it as a string. *large_object* specifies a valid large object resource and *len* specifies the maximum allowable size of the large object segment. It returns FALSE if there is an error.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called `pg_loread()`.

See also pg_lo_read_all().

pg_lo_read_all (PHP 4 >= 4.2.0)

Read a entire large object and send straight to browser

int **pg_lo_read_all** (resource large_object) \linebreak

pg_lo_read_all() reads a large object and passes it straight through to the browser after sending all pending headers. Mainly intended for sending binary data like images or sound. It returns number of bytes read. It returns `FALSE`, if an error occurred.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called `pg_loreadall()`.

See also pg_lo_read().

pg_lo_seek (PHP 4 >= 4.2.0)

Seeks position of large object

bool **pg_lo_seek** (resource large_object, int offset [, int whence]) \linebreak

pg_lo_seek() seeks position of large object resource. *`whence`* is PGSQL_SEEK_SET, PGSQL_SEEK_CUR or PGSQL_SEEK_END.

See also pg_lo_tell().

pg_lo_tell (PHP 4 >= 4.2.0)

Returns current position of large object

int **pg_lo_tell** (resource large_object) \linebreak

pg_lo_tell() returns current position (offset from the beginning of large object).

See also pg_lo_seek().

pg_lo_unlink (PHP 4 >= 4.2.0)

Delete a large object

bool **pg_lo_unlink** (resource connection, int oid) \linebreak

pg_lo_unlink() deletes a large object with the *`oid`*. It returns `TRUE` on success, otherwise returns `FALSE`.

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called `pg_lo_unlink()`.

See also pg_lo_create() and pg_lo_import().

pg_lo_write (PHP 4 >= 4.2.0)

Write a large object

int **pg_lo_write** (resource large_object, string data) \linebreak

pg_lo_write() writes at most to a large object from a variable *data* and returns the number of bytes actually written, or FALSE in the case of an error. *large_object* is a large object resource from pg_lo_open().

To use the large object (lo) interface, it is necessary to enclose it within a transaction block.

> **Nota:** This function used to be called `pg_lo_write()`.

See also pg_lo_create() and pg_lo_open().

pg_metadata (PHP 4 CVS only)

Get metadata for table.

array **pg_metadata** (resource connection, string table_name) \linebreak

pg_metadata() returns table definition for `table_name` as array. If there is error, it returns FALSE

> **Nota:** This function is experimental.

See also pg_convert()

pg_num_fields (PHP 4 >= 4.2.0)

Returns the number of fields

int **pg_num_fields** (resource result) \linebreak

pg_num_fields() returns the number of fields (columns) in a PostgreSQL *result*. The argument is a result resource returned by pg_query(). This function will return -1 on error.

> **Nota:** This function used to be called `pg_numfields()`.

See also pg_num_rows() and pg_affected_rows().

pg_num_rows (PHP 4 >= 4.2.0)

Returns the number of rows

int **pg_num_rows** (resource result) \linebreak

pg_num_rows() will return the number of rows in a PostgreSQL *result* resource. *result* is a query result resource returned by pg_query(). This function will return -1 on error.

> **Nota:** Use pg_affected_rows() to get number of rows affected by INSERT, UPDATE and DELETE query.

> **Nota:** This function used to be called `pg_numrows()`.

See also pg_num_fields() and pg_affected_rows().

pg_options (PHP 3, PHP 4 >= 4.0.0)

Get the options associated with the connection

string **pg_options** (resource connection) \linebreak

pg_options() will return a string containing the options specified on the given PostgreSQL *connection* resource.

pg_pconnect (PHP 3, PHP 4 >= 4.0.0)

Open a persistent PostgreSQL connection

int **pg_pconnect** (string connection_string) \linebreak

pg_pconnect() opens a connection to a PostgreSQL database. It returns a connection resource that is needed by other PostgreSQL functions.

It returns a connection resource on success, or `FALSE` if the connection could not be made. The arguments should be within a quoted string. The arguments available include *host*, *port*, *tty*, *options*, *dbname*, *user*, and *password*.

Esempio 1. Using pg_pconnect

```
<?php
$dbconn = pg_connect ("dbname=mary");
//connect to a database named "mary"
$dbconn2 = pg_connect ("host=localhost port=5432 dbname=mary");
// connect to a database named "mary" on "localhost" at port "5432"
$dbconn3 = pg_connect ("host=sheep port=5432 dbname=mary user=lamb password=foo");
//connect to a database named "mary" on the host "sheep" with a username and password
```

```
$conn_string = "host=sheep port=5432 dbname=test user=lamb password=bar";
$dbconn4 = pg_connect ($conn_string);
//connect to a database named "test" on the host "sheep" with a username and password
?>
```

If a second call is made to **pg_pconnect()** with the same arguments, no new connection will be established, but instead, the connection resource of the already opened connection will be returned. You can have multiple connections to the same database if you use different connection string.

Multiple parameters syntax for **pg_pconnect() $conn = pg_pconnect ("host", "port", "options", "tty", "dbname")** has been deprecated.

To enable persistent connection, pgsql.allow_persistent `php.ini` directive must be set to "On". (Default is On) Max number of persistent connection can be defined by pgsql.max_persistent `php.ini` directive. (Default is -1 which is no limit) Total number of connection can be set by pgsql.max_links `php.ini` directive.

pg_close() will not close persistent links generated by **pg_pconnect()**.

See also pg_connect().

pg_port (PHP 3, PHP 4 >= 4.0.0)

Return the port number associated with the connection

int **pg_port** (resource connection) \linebreak

pg_port() returns the port number that the given PostgreSQL *connection* resource is connected to.

pg_put_line (PHP 4 >= 4.0.3)

Send a NULL-terminated string to PostgreSQL backend

bool **pg_put_line** ([resource connection, string data]) \linebreak

pg_put_line() sends a NULL-terminated string to the PostgreSQL backend server. This is useful for example for very high-speed inserting of data into a table, initiated by starting a PostgreSQL copy-operation. That final NULL-character is added automatically. It returns `TRUE` if successful, `FALSE` otherwise.

> **Nota:** Note the application must explicitly send the two characters "\." on a final line to indicate to the backend that it has finished sending its data.

See also pg_end_copy().

Esempio 1. High-speed insertion of data into a table

```
<?php
    $conn = pg_pconnect ("dbname=foo");
    pg_exec($conn, "create table bar (a int4, b char(16), d float8)");
    pg_exec($conn, "copy bar from stdin");
    pg_put_line($conn, "3\thello world\t4.5\n");
    pg_put_line($conn, "4\tgoodbye world\t7.11\n");
    pg_put_line($conn, "\\.\n");
    pg_end_copy($conn);
?>
```

pg_query (PHP 4 >= 4.2.0)

Execute a query

resource **pg_query** (resource connection, string query) \linebreak

pg_query() returns a query result resource if query could be executed. It returns `FALSE` on failure or if connection is not a valid connection. Details about the error can be retrieved using the pg_last_error() function if connection is valid. pg_last_error() sends an SQL statement to the PostgreSQL database specified by the *connection* resource. The *connection* must be a valid connection that was returned by pg_connect() or pg_pconnect(). The return value of this function is an query result resource to be used to access the results from other PostgreSQL functions such as pg_fetch_array().

> **Nota:** *connection* is a optional parameter for **pg_query()**. If *connection* is not set, default connection is used. Default connection is the last connection made by pg_connect() or pg_pconnect().
>
> Although *connection* can be omitted, it is not recommended, since it could be a cause of hard to find bug in script.

> **Nota:** This function used to be called `pg_exec()`. `pg_exec()` is still available for compatibility reasons but users are encouraged to use the newer name.

See also pg_connect(), pg_pconnect(), pg_fetch_array(), pg_fetch_object(), pg_num_rows(), and pg_affected_rows().

pg_result_error (PHP 4 >= 4.2.0)

Get error message associated with result

string **pg_result_error** (resource result) \linebreak

pg_result_error() returns error message associated with *result* resource. Therefore, user has better chance to get better error message than pg_last_error().

See also pg_query(), pg_send_query(), pg_get_result(), pg_last_error() and pg_last_notice()

pg_result_status (PHP 4 >= 4.2.0)

Get status of query result

int **pg_result_status** (resource result) \linebreak

pg_result_status() returns status of result resource. Possible retun values are PGSQL_EMPTY_QUERY, PGSQL_COMMAND_OK, PGSQL_TUPLES_OK, PGSQL_COPY_TO, PGSQL_COPY_FROM, PGSQL_BAD_RESPONSE, PGSQL_NONFATAL_ERROR and PGSQL_FATAL_ERROR.

See also pg_connection_status().

pg_select (PHP 4 CVS only)

Select records.

array **pg_select** (resource connection, string table_name, array assoc_array [, int options]) \linebreak

pg_select() selects records specified by `assoc_array` which has `field=>value`. For successful query, it returns array contains all records and fields that match the condition specified by `assoc_array`. If `options` is specified, pg_convert() s applied to `assoc_array` with specified option.

Esempio 1. pg_select

```
<?php
    $db = pg_connect ('dbname=foo');
    // This is safe, since $_POST is converted autotmatically
    $rec = pg_select($db, 'post_log', $_POST);
    if ($rec) {
        echo "Records selected\n";
        var_dump($rec);
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Nota: This function is experimental.

See also pg_convert()

pg_send_query (PHP 4 >= 4.2.0)

Send asynchronous query

bool **pg_send_query** (resource connection, string query) \linebreak bool **pg_send_query** (string query) \linebreak

pg_send_query() send asynchronous query to the *connection*. Unlike pg_query(), it can send multiple query to PostgreSQL and get the result one by one using pg_get_result(). Script execution is not block while query is executing. Use pg_connection_busy() to check connection is busy (i.e. query is executing) Query may be canceled by calling pg_cancel_query().

Although, user can send multiple query at once. User cannot send multiple query over busy connection. If query is sent while connection is busy, it waits until last query is finished and discards all result.

See also pg_query(), pg_cancel_query(), pg_get_result() and pg_connection_busy()

pg_set_client_encoding (PHP 3 CVS only, PHP 4 >= 4.0.3)

Set the client encoding

int **pg_set_client_encoding** ([resource connection, string encoding]) \linebreak

pg_set_client_encoding() sets the client encoding and return 0 if success or -1 if error.

encoding is the client encoding and can be either : SQL_ASCII, EUC_JP, EUC_CN, EUC_KR, EUC_TW, UNICODE, MULE_INTERNAL, LATINX (X=1...9), KOI8, WIN, ALT, SJIS, BIG5, WIN1250. Available encoding depends on your PostgreSQL and libpq version. Refer to PostgreSQL manual for supported encodings for your PostgreSQL.

> **Nota:** This function requires PHP-4.0.3 or higher and PostgreSQL-7.0 or higher. Supported encoding depends on PostgreSQL version. Refer to PostgreSQL manual for details.
>
> The function used to be called **pg_setclientencoding()**.

See also pg_client_encoding().

pg_trace (PHP 4)

Enable tracing a PostgreSQL connection

bool **pg_trace** (string pathname [, string mode [, resource connection]]) \linebreak

pg_trace() enables tracing of the PostgreSQL frontend/backend communication to a debugging file specified as *pathname*. To fully understand the results, one needs to be familiar with the internals of PostgreSQL communication protocol. For those who are not, it can still be useful for tracing

errors in queries sent to the server, you could do for example **grep '^To backend' trace.log** and see what query actually were sent to the PostgreSQL server. For more information, refer to PostgreSQL manual.

Filename and *mode* are the same as in fopen() (*mode* defaults to 'w'), *connection* specifies the connection to trace and defaults to the last one opened.

It returns TRUE if *pathname* could be opened for logging, FALSE otherwise.

See also fopen() and pg_untrace().

pg_tty (PHP 3, PHP 4 >= 4.0.0)

Return the tty name associated with the connection

string **pg_tty** (resource connection) \linebreak

pg_tty() returns the tty name that server side debugging output is sent to on the given PostgreSQL *connection* resource.

pg_untrace (PHP 4)

Disable tracing of a PostgreSQL connection

bool **pg_untrace** ([resource connection]) \linebreak

Stop tracing started by pg_trace(). *connection* specifies the connection that was traced and defaults to the last one opened.

Returns always TRUE.

See also pg_trace().

pg_update (PHP 4 CVS only)

Update table.

long **pg_update** (resource connection, string table_name, array condition, array data [, int options]) \linebreak

pg_update() updates records that matches condition with data If options is specified, pg_convert() is applied to assoc_array with specified options.

Esempio 1. pg_update

```
<?php
    $db = pg_connect ('dbname=foo');
    $data = array('field1'=>'AA', 'field2'=>'BB');
    // This is safe, since $_POST is converted automatically
    $res = pg_update($db, 'post_log', $_POST, $data);
```

```
    if ($res) {
        echo "Data is updated: $res\n";
    }
    else {
        echo "User must have sent wrong inputs\n";
    }
?>
```

Nota: This function is experimental.

See also pg_convert()

LXXX. Process Control Functions

Process Control support in PHP implements the Unix style of process creation, program execution, signal handling and process termination. Process Control should not be enabled within a webserver environment and unexpected results may happen if any Process Control functions are used within a webserver environment.

This documentation is intended to explain the general usage of each of the Process Control functions. For detailed information about Unix process control you are encouraged to consult your systems documentation including fork(2), waitpid(2) and signal(2) or a comprehensive reference such as Advanced Programming in the UNIX Environment by W. Richard Stevens (Addison-Wesley).

Process Control support in PHP is not enabled by default. You will need to use the --enable-pcntl configuration option when compiling PHP to enable Process Control support.

> **Nota:** Currently, this module will not function on non-Unix platforms (Windows).

The following list of signals are supported by the Process Control functions. Please see your systems signal(7) man page for details of the default behavior of these signals.

Tabella 1. Supported Signals

SIGFPE	SIGCONT	SIGKILL
SIGSTOP	SIGUSR1	SIGTSTP
SIGHUP	SIGUSR2	SIGTTIN
SIGINT	SIGSEGV	SIGTTOU
SIGQUIT	SIGPIPE	SIGURG
SIGILL	SIGALRM	SIGXCPU
SIGTRAP	SIGTERM	SIGXFSZ
SIGABRT	SIGSTKFLT	SIGVTALRM
SIGIOT	SIGCHLD	SIGPROF
SIGBUS	SIGCLD	SIGWINCH
SIGPOLL	SIGIO	SIGPWR
SIGSYS		

Process Control Example

This example forks off a daemon process with a signal handler.

Esempio 1. Process Control Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
```

```
     die("could not fork");
} else if ($pid) {
     exit(); // we are the parent
} else {
     // we are the child
}

// detatch from the controlling terminal
if (!posix_setsid()) {
    die("could not detach from terminal");
}

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");

// loop forever performing tasks
while(1) {

    // do something interesting here

}

function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         default:
             // handle all other signals
     }

}

?>
```

pcntl_exec (PHP 4 >= 4.2.0)

Executes specified program in current process space

bool **pcntl_exec** (string path [, array args [, array envs]]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

pcntl_fork (PHP 4 >= 4.1.0)

Forks the currently running process

int **pcntl_fork** (void) \linebreak

The **pcntl_fork()** function creates a child process that differs from the parent process only in it's PID and PPID. Please see your system's fork(2) man page for specific details as to how fork works on your system.

On success, the PID of the child process is returned in the parent's thread of execution, and a 0 is returned in the child's thread of execution. On failure, a -1 will be returned in the parent's context, no child process will be created, and a PHP error is raised.

Esempio 1. pcntl_fork() Example

```
<?php

$pid = pcntl_fork();
if ($pid == -1) {
     die("could not fork");
} else if ($pid) {
     // we are the parent
} else {
     // we are the child
}

?>
```

See also pcntl_waitpid() and pcntl_signal().

pcntl_signal (PHP 4 >= 4.1.0)

Installs a signal handler

bool **pcntl_signal** (int signo, mixed handle) \linebreak

The **pcntl_signal()** function installs a new signal handler for the signal indicated by *signo*. The signal handler is set to *handler* which may be the name of a user created function, or either of the two global constants SIG_IGN or SIG_DFL.

pcntl_signal() returns TRUE on success or FALSE on failure.

Esempio 1. pcntl_signal() Example

```
<?php

// signal handler function
function sig_handler($signo) {

     switch($signo) {
         case SIGTERM:
             // handle shutdown tasks
             exit;
             break;
         case SIGHUP:
             // handle restart tasks
             break;
         case SIGUSR1:
             print "Caught SIGUSR1...\n";
             break;
         default:
             // handle all other signals
     }

}

print "Installing signal handler...\n";

// setup signal handlers
pcntl_signal(SIGTERM, "sig_handler");
pcntl_signal(SIGHUP, "sig_handler");
pcntl_signal(SIGUSR1, "sig_handler");

print "Generating signal SIGTERM to self...\n";

// send SIGUSR1 to current process id
posix_kill(posix_getpid(), SIGUSR1);

print "Done\n"

?>
```

See also pcntl_fork() and pcntl_waitpid().

pcntl_waitpid (PHP 4 >= 4.1.0)

Waits on or returns the status of a forked child

int **pcntl_waitpid** (int pid, int status, int options) \linebreak

The **pcntl_waitpid()** function suspends execution of the current process until a child as specified by the *pid* argument has exited, or until a signal is delivered whose action is to terminate the current process or to call a signal handling function. If a child as requested by *pid* has already exited by the time of the call (a so-called "zombie" process), the function returns immediately. Any system resources used by the child are freed. Please see your system's waitpid(2) man page for specific details as to how waitpid works on your system.

pcntl_waitpid() returns the process ID of the child which exited, -1 on error or zero if WNOHANG was used and no child was available

The value of *pid* can be one of the following:

Tabella 1. possible values for *pid*

`< -1`	wait for any child process whose process group ID is equal to the absolute value of *pid*.
`-1`	wait for any child process; this is the same behaviour that the wait function exhibits.
`0`	wait for any child process whose process group ID is equal to that of the calling process.
`> 0`	wait for the child whose process ID is equal to the value of *pid*.

pcntl_waitpid() will store status information in the *status* parameter which can be evaluated using the following functions: pcntl_wifexited(), pcntl_wifstopped(), pcntl_wifsignaled(), pcntl_wexitstatus(), pcntl_wtermsig() and pcntl_wstopsig().

The value of *options* is the value of zero or more of the following two global constants `OR`'ed together:

Tabella 2. possible values for *options*

`WNOHANG`	return immediately if no child has exited.
`WUNTRACED`	return for children which are stopped, and whose status has not been reported.

See also pcntl_fork(), pcntl_signal(), pcntl_wifexited(), pcntl_wifstopped(), pcntl_wifsignaled(), pcntl_wexitstatus(), pcntl_wtermsig() and pcntl_wstopsig().

pcntl_wexitstatus (PHP 4 >= 4.1.0)

Returns the return code of a terminated child

int **pcntl_wexitstatus** (int status) \linebreak

Returns the return code of a terminated child. This function is only useful if pcntl_wifexited() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wifexited().

pcntl_wifexited (PHP 4 >= 4.1.0)

Returns `TRUE` if status code represents a successful exit

int **pcntl_wifexited** (int status) \linebreak

Returns `TRUE` if the child status code represents a successful exit.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wexitstatus().

pcntl_wifsignaled (PHP 4 >= 4.1.0)

Returns `TRUE` if status code represents a termination due to a signal

int **pcntl_wifsignaled** (int status) \linebreak

Returns `TRUE` if the child process exited because of a signal which was not caught.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_signal().

pcntl_wifstopped (PHP 4 >= 4.1.0)

Returns `TRUE` if child process is currently stopped

int **pcntl_wifstopped** (int status) \linebreak

Returns `TRUE` if the child process which caused the return is currently stopped; this is only possible if the call to pcntl_waitpid() was done using the option `WUNTRACED`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid().

pcntl_wstopsig (PHP 4 >= 4.1.0)

Returns the signal which caused the child to stop

int **pcntl_wstopsig** (int status) \linebreak

Returns the number of the signal which caused the child to stop. This function is only useful if pcntl_wifstopped() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid() and pcntl_wifstopped().

pcntl_wtermsig (PHP 4 >= 4.1.0)

Returns the signal which caused the child to terminate

int **pcntl_wtermsig** (int status) \linebreak

Returns the number of the signal that caused the child process to terminate. This function is only useful if pcntl_wifsignaled() returned `TRUE`.

The parameter *`status`* is the status parameter supplied to a successfull call to pcntl_waitpid().

See also pcntl_waitpid(), pcntl_signal() and pcntl_wifsignaled().

LXXXI. Funzioni per l’esecuzione di programmi

Queste funzioni permettono l’esecuzione di comandi sul sistema stesso, e rendono sicuri tali comandi. Queste funzioni sono anche strettamente correlate al operatore backtick.

escapeshellarg (PHP 4 >= 4.0.3)

Estrae una stringa da usare come un argomento della shell

string **escapeshellarg** (string arg) \linebreak

escapeshellarg() aggiunge le virgolette singole attorno ad una stringa ed elude ogni virgoletta semplice per permetterti di passare una stringa direttamente ad una funzione della shell a che questa venga trattata come un singolo argomento. Questa funzione dovrebbe essere usata per eludere argomenti individuali per funzioni della shell che giungano dall'input del utente. Le funzioni della shell includono exec(), system() e l'operatore backtick. Un utilizzo standard sarebbe:

```
system("ls ".escapeshellarg($dir));
```

Vedere anche exec(), popen(), system() e l'operatore backtick.

escapeshellcmd (PHP 3, PHP 4 >= 4.0.0)

Elude i metacaratteri della shell

string **escapeshellcmd** (string command) \linebreak

escapeshellcmd() elude ogni carattere di una stringa che potrebbe essere usata per indurre un comando shell ad eseguire comandi arbitrari. Questa funzione dovrebbe essere usata per assicurarsi che ogni dato che giunga dall'input dell'utente venga neutralizzato prima di essere passato a funzioni come exec() o system() o all'operatore backtick . Un modello d'utilizzo potrebbe essere:

```
$e = escapeshellcmd($userinput);
system("echo $e"); // qui non ci preoccupiamo se $e contiene spazi
$f = escapeshellcmd($filename);
system("touch \"/tmp/$f\"; ls -l \"/tmp/$f\""); // e qui invece lo facciamo, us-
ando le virgolette
```

Vedere anche escapeshellarg(), exec(), popen(), system() e l'operatore backtick.

exec (PHP 3, PHP 4 >= 4.0.0)

Esegue un programma esterno

string **exec** (string command [, string array [, int return_var]]) \linebreak

exec() esegue il comando passato da *`command`*, sebbene esso non invii nulla in output Restituisce semplicemente l'ultima linea dal risultato del comando. Se hai bisogno di eseguire un comando ed avere tutti i dati dal comando passato direttamente senza alcuna interferenza, usa la funzione passthru().

Se l'argomento *`array`* è presente, allora tale vettore specificato verrà riempito con ogni linea del output del comando. Nota che se il vettore contiene già degli elementi, **exec()** li aggiungerà in coda vettore. Se non si vuole che la funzione aggiunga elementi, eseguire un unset() sul vettore prima di passarlo ad **exec()**.

Se viene passato l'argomento *`return_var`* assieme all'argomento *`array`*, allora lo stato del comando eseguito verra scitto in questa variabile.

> **Attenzione**
>
> Osserva che se intendi allocare dati che giungano dal utente per essere passate a questa funzione, dovresti usare escapeshellarg() o escapeshellcmd() per assicurarti che l'utente non possa forzare il sistema ad eseguire comandi arbitrari.

Nota: Nota anche che se inizi un programma usando questa funzione e lo vuoi lasciare in esecuzione in background, devi assicurarti che l'output del programa venga rediretto ad un file o a qualche altro flusso di output, altrimenti PHP attendera fino alla fine fine dell'esecuzione del programma.

Vedere anche system(), passthru(), popen(), escapeshellcmd() e l'operatore backtick.

passthru (PHP 3, PHP 4 >= 4.0.0)

Esegue un programma esterno e mostra l'output non elaborato

void **passthru** (string command [, int return_var]) \linebreak

La funzione **passthru()** è simile alla funzione exec() inquanto esegue *`command`*. Se il parametro *`return_var`* è specificato, lo stato ritornato dal comando Unix verrà posto lì. Questa funzione deve essere usata al posto di exec() o di system() quando l'output del comando Unix consiste in dati binari da passare direttamente al browser. Un suo uso frequente consiste nel eseguire, ad esempio, le utility pbmplus che possono restituire un flusso diretto all'immagine. Impostado il tipo di contenuto a *image/gif* e successivamente chiamando un programma pbmplus per generare una gif puoi realizzare uno script PHP che genera direttamente immagini.

> **Attenzione**
>
> Se si intende permettere ai dati provenienti dall'input dell'utente di essere passati a questa funzione, allora si dovrebbe usare escapeshellarg() o escapeshellcmd(), questo al fine di assicurarsi che gli utenti non possano maliziosamente indurrethe il sistema ad eseguire comandi arbitrari.

> **Nota:** Nota che se avvii un programma usando questa funzione e intendi lasciarlo girare in background, devi accertarti che l'output del programma sia rediretto in un file o a qualche altro flusso di output o PHP attenderà fino alla fine dell'esecuzione del programma.

Vedere anche exec(), system(), popen(), escapeshellcmd(), e l'operatore backtick.

proc_close (PHP 4 CVS only)

Close a process opened by proc_open and return the exit code of that process.

int **proc_close** (resource process) \linebreak

proc_close() is similar to popen() except that it only works on processes opened by proc_open(). **proc_close()** waits for the process to terminate, and returns it's exit code. If you have open pipes to that process, you should fclose() them prior to calling this function in order to avoid a deadlock - the child process may not be able to exit while the pipes are open.

proc_open (PHP 4 CVS only)

Execute a command and open file pointers for input/output

resource **proc_open** (string cmd, array descriptorspec, array pipes) \linebreak

proc_open() is similar to popen() but provides a much greater degree of control over the program execution. `cmd` is the command to be executed by the shell. `descriptorspec` is an indexed array where the key represents the descriptor number and the value represents how PHP will pass that descriptor to the child process. `pipes` will be set to an indexed array of file pointers that correspond to PHP's end of any pipes that are created. The return value is a resource representing the process; you should free it using proc_close() when you are finished with it.

```
$descriptorspec = array(
   0 => array("pipe", "r"),  // stdin is a pipe that the child will read from
   1 => array("pipe", "w"),  // stdout is a pipe that the child will write to
   2 => array("file", "/tmp/error-output.txt", "a"), // stderr is a file to write to
);
$process = proc_open("php", $descriptorspec, $pipes);
if (is_resource($process)) {
    // $pipes now looks like this:
    // 0 => writeable handle connected to child stdin
    // 1 => readable handle connected to child stdout
    // Any error output will be appended to /tmp/error-output.txt

    fwrite($pipes[0], "<?php echo \"Hello World!\"; ?>");
    fclose($pipes[0]);

    while(!feof($pipes[1])) {
        echo fgets($pipes[1], 1024);
    }
```

```
    fclose($pipes[1]);
    // It is important that you close any pipes before calling
    // proc_close in order to avoid a deadlock
    $return_value = proc_close($process);

    echo "command returned $return_value\n";
}
```

The file descriptor numbers in *descriptorspec* are not limited to 0, 1 and 2 - you may specify any valid file descriptor number and it will be passed to the child process. This allows your script to interoperate with other scripts that run as "co-processes". In particular, this is useful for passing passphrases to programs like PGP, GPG and openssl in a more secure manner. It is also useful for reading status information provided by those programs on auxillary file descriptors.

> **Nota:** Windows compatibility: Descriptors beyond 2 (stderr) are made available to the child process as inheritable handles, but since the Windows architecture does not associate file descriptor numbers with low-level handles, the child process does not (yet) have a means of accessing those handles. Stdin, stdout and stderr work as expected.

> **Nota:** This function was introduced in PHP 4.3.0.

> **Nota:** If you only need a uni-directional (one-way) process pipe, use popen() instead, as it is much easier to use.

See also exec(), system(), passthru(), popen(), escapeshellcmd(), and the backtick operator.

shell_exec (PHP 4 >= 4.0.0)

Esegue un comando attraverso la shell e restituisce l'output come stringa

string **shell_exec** (string cmd) \linebreak

Questa funzione è identica all'operatore backtick.

system (PHP 3, PHP 4 >= 4.0.0)

Esegue un programma esterno e mostra l'output

string **system** (string command [, int return_var]) \linebreak

system() è semplicemente come la versione C della funzione che esegue il *`command`* dato e restituisce in uscita il risultato. Se viene fornita una variabile come secondo argomento, allora il codice di stato ritornato dal comando eseguito verrà scritto in tale variabile.

Attenzione

Nota che se intendi permettere che i dati in ingresso dall'utente vengano passati a questa funzione, dovresti utilizzare la funzione escapeshellarg() o escapeshellcmd() per assicurarti che l'utente non possa forzare il sistema ad eseguire comandi arbitrari.

Nota: Nota che se avvii un programma usando questa funzione e intendi lasciarlo girare in background, devi accertarti che l'output del programma sia rediretto in un file o a qualche altro flusso di output o PHP attenderà fino alla fine dell'esecuzione del programma.

La chiamata a **system()** tenta anche di ripulire automaticamente il buffer di output del web server dopo ogni linea di output se PHP gira come un modulo server.

Restituisce l'ultima linea del output del comando se ha successo e `FALSE` se fallisce.

Se devi eseguire un comando ottenendo tutti i dati restituiti dal comando direttamente senza alcuna interferenza, usa la funzione passthru().

Vedere anche exec(), passthru(), popen(), escapeshellcmd() e l'operatore backtick.

LXXXII. Funzioni per le stampanti

Queste funzioni sono disponibili solo nei sistemi Windows 9.x, ME, NT4 e 2000. Sono state aggiunte in PHP 4 (4.0.4).

printer_abort (unknown)

Cancella il file di spool dalla stampante

void **printer_abort** (resource handle) \linebreak

La funzione cancella lo spool di stampa dalla stampante.

Il parametro *handle* deve indicare un handle valido di stampante.

Esempio 1. Esempio di utilizzo di printer_abort()

```
$handle = printer_open();
printer_abort($handle);
printer_close($handle);
```

printer_close (unknown)

Chiude una connessione aperta verso una stampante

void **printer_close** (resource handle) \linebreak

Questa funzione chiude la connessione verso una stampante. Inoltre, la funzione **printer_close()** chiude il device context attivo.

Il parametro *handle* deve indicare un handle valido di stampante.

Esempio 1. Esempio di utilizzo di printer_close()

```
$handle = printer_open();
printer_close($handle);
```

printer_create_brush (unknown)

Crea un nuovo pennello

mixed **printer_create_brush** (int stile, string colore) \linebreak

La funzione crea un nuovo pennello e restituisce un handle per questo. Il pennello viene usato per riempire le forme. Per un esempio vedere printer_select_brush(). Il parametro *colore* deve essere un colore in formato esadecimale RGB, ad esempio "000000" per il nero, *stile*, invece, deve essere valorizzato con una delle seguenti costanti:

- *PRINTER_BRUSH_SOLID*: crea un pennello con un colore pieno.

- *`PRINTER_BRUSH_DIAGONAL`*: crea un pennello con righe a 45 gradi dal basso a sinistra verso l'alto a destra (/).
- *`PRINTER_BRUSH_CROSS`*: crea un pennello con delle croci (+).
- *`PRINTER_BRUSH_DIAGCROSS`*: crea un pennello con croci a 45 gradi (x).
- *`PRINTER_BRUSH_FDIAGONAL`*: crea un pennello con righe a 45 gradi dall'alto a sinistra verso il basso a destra (\ ).
- *`PRINTER_BRUSH_HORIZONTAL`*: crea un pennello a righe orizzontali (-).
- *`PRINTER_BRUSH_VERTICAL`*: crea un pennello a righe verticali (|).
- *`PRINTER_BRUSH_CUSTOM`*: crea un pennello personalizzato a partire da un file BMP. Il secondo parametro viene usato per indicare il file BMP anzichè il codice RGB del colore.

printer_create_dc (unknown)

Crea un nuovo device context

void **printer_create_dc** (resource handle) \linebreak

La funzione crea un nuovo device context. Il device context è utilizzato per personalizzare gli oggetti grafici del documento. Il parametro *`handle`* deve indicare un handle valido di stampante.

Esempio 1. Esempio di utilizzo di printer_create_dc()

```
$handle = printer_open();
printer_start_doc($handle);
printer_start_page($handle);

printer_create_dc($handle);
/* esegue qualche operazione sul device context */
printer_set_option($handle, PRINTER_TEXT_COLOR, "333333");
printer_draw_text($handle, 1, 1, "text");
printer_delete_dc($handle);

/* crea un'altro device context */
printer_create_dc($handle);
printer_set_option($handle, PRINTER_TEXT_COLOR, "000000");
printer_draw_text($handle, 1, 1, "text");
/* ci esegue delle operazioni */

printer_delete_dc($handle);

printer_endpage($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_create_font (unknown)

Crea un nuovo font

mixed **printer_create_font** (string tipo, int altezza, int larghezza, int spessore, bool corsivo, bool sottolineato, bool barrato, int orientamento) \linebreak

La funzione crea un nuovo font e restituisce il relativo handle. Il font è utilizzato per scrivere testi. Per un esempio vedere printer_select_font(). Il parametro *`tipo`* è una stringa indicante il tipo di font. *`Altezza`* indica l'altezza del font e *`larghezza`* ne indica la larghezza. Il parametro *`spessore`* indica lo spessore del font (il valore normale è 400), e può essere una delle seguente costanti predefinite:

- *`PRINTER_FW_THIN`*: imposta un font sottile (100).
- *`PRINTER_FW_ULTRALIGHT`*: imposta un font molto leggero (200).
- *`PRINTER_FW_LIGHT`*: imposta un font leggero (300).
- *`PRINTER_FW_NORMAL`*: imposta un font normale (400).
- *`PRINTER_FW_MEDIUM`*: imposta un font medio (500).
- *`PRINTER_FW_BOLD`*: imposta il font a grassetto (700).
- *`PRINTER_FW_ULTRABOLD`*: imposta il font ad un grassetto maggiore (800).
- *`PRINTER_FW_HEAVY`*: imposta un font grosso (900).

Il parametro *`corsivo`* può essere `TRUE` o `FALSE`, ed indica se il font debba essere corsivo.

Il parametro *`sottolineato`* può essere `TRUE` o `FALSE`, e indica se il font debba essere sottolineato.

Il parametro *`barrato`* può essere `TRUE` o `FALSE`, e indica se il font debba essere barrato.

Il parametro *`orientamento`* specifica la rotazione. Per un esempio vedere printer_select_font().

printer_create_pen (unknown)

Crea una nuova penna

mixed **printer_create_pen** (int stile, int larghezza, string colore) \linebreak

La funzione crea una nuova penna e restituisce il relativo handle. Una penna viene usata per tracciare linee e curve. Per un esempio vedere printer_select_pen(). Il parametro *`colore`* deve essere un colore in formato esadecimale RGB, ad esempio "000000" per nero, *`larghezza`* specifica la larghezza della penna, mentre *`stile`* deve essere una delle seguenti costanti:

- *`PRINTER_PEN_SOLID`*: crea una penna continua.
- *`PRINTER_PEN_DASH`*: crea una penna tratteggiata.
- *`PRINTER_PEN_DOT`*: crea una penna a punti.
- *`PRINTER_PEN_DASHDOT`*: crea una penna con tratto e punto.
- *`PRINTER_PEN_DASHDOTDOT`*: crea una penna con tratto e due punti.
- *`PRINTER_PEN_INVISIBLE`*: crea una penna invisibile.

printer_delete_brush (unknown)

Cancella un pennello

bool **printer_delete_brush** (resource handle) \linebreak

La funzione cancella il pennello selezionato. Per un esempio vedere printer_select_brush(). La funzione restituisce `TRUE` se riesce, oppure `FALSE` in caso contrario. Il parametro *`handle`* deve essere un handle valido di un pennello.

printer_delete_dc (unknown)

Cancella un device context

bool **printer_delete_dc** (resource handle) \linebreak

Questa funzione cancella il device context e restituisce `TRUE` se ha successo, oppure `FALSE` se si verifica un errore. Per un esempio vedere printer_create_dc(). Il parametro *`handle`* deve essere un handle valido di stampante.

printer_delete_font (unknown)

Cancella un font

bool **printer_delete_font** (resource handle) \linebreak

La funzione cancella il font prescelto. Per un esempio vedere printer_select_font(). La funzione restituisce `TRUE` se ha successo, oppure `FALSE`. Il parametro *`handle`* deve essere un handle valido di un font.

printer_delete_pen (unknown)

Cancella una penna

bool **printer_delete_pen** (resource handle) \linebreak

La funzione cancella la penna prescelta. Per un esempio vedere printer_select_pen(). La funzione restituisce `TRUE` se ha successo, oppure `FALSE` se si verifica un errore. Il parametro *`handle`* deve essere un handle valido di una penna.

printer_draw_bmp (unknown)

Disegna una bitmap

void **printer_draw_bmp** (resource handle, string nomefile, int x, int y) \linebreak

La funzione disegna la bitmap `nomefile` alla posizione `x`, `y`. Il parametro `handle` deve essere un handle valido di una stampante.

La funzione restituisce `TRUE` se ha successo, oppure `FALSE` in caso contrario.

Esempio 1. Esempio di utilizzo di printer_draw_bmp()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_draw_bmp($handle, "c:\\image.bmp", 1, 1);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_chord (unknown)

Traccia una corda

void **printer_draw_chord** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad_x, int rad_y, int rad_x1, int rad_y1) \linebreak

La funzione disegna una corda. Il parametro `handle` deve esesre un handle valido di stampante.

`rec_x` indica la coordinata x dell'angolo in alto a sinistra del rettangolo perimetrale.

`rec_y` indica la coordinata y dell'angolo in alto a sinistra del rettangolo perimetrale.

`rec_x1` indica la coordinata x dell'angolo in basso a destra del rettangolo perimetrale.

`rec_y1` indica la coordinata y dell'angolo in basso a destra del rettangolo perimetrale.

`rad_x` indica la coordinata x del radiale indicante l'inizio della corda.

`rad_y` indica la coordinata y del radiale indicante l'inizio della corda.

`rad_x1` indica la coordinata x del radiale indicante la fine della corda.

`rad_y1` indica la coordinata y del radiale indicante la fine della corda.

Esempio 1. Esempio di utilizzo di printer_draw_chord()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);
```

```
$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_chord($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_elipse (unknown)

Traccia una ellisse

void **printer_draw_elipse** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y) \linebreak

La funzione disegna una ellisse. Il parametro `handle` deve esesre un handle valido di stampante.

`ul_x` indica la coordinata x in alto a sinistra dell'ellisse.

`ul_y` indica la coordinata y in alto a sinistra dell'ellisse.

`lr_x` indica la coordinata x in basso a destra dell'ellisse.

`lr_y` indica la coordinata y in basso a destra dell'ellisse.

Esempio 1. Esempio di utilizzo di printer_draw_elipse()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_elipse($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_line (unknown)

Traccia una linea

void **printer_draw_line** (resource printer_handle, int from_x, int from_y, int to_x, int to_y) \linebreak

Questa funzione disegna una linea dalla posizione *from_x*, *from_y* alla posizione *to_x*, *to_y* usando la penna selezionata. Il parametro *handle* deve essere un handle valido di stampante.

Esempio 1. Esempio di utilizzo di printer_draw_line()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "000000");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 10, 1000, 10);
printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_pie (unknown)

Traccia una grafico a torta

void **printer_draw_pie** (resource handle, int rec_x, int rec_y, int rec_x1, int rec_y1, int rad1_x, int rad1_y, int rad2_x, int rad2_y) \linebreak

La funzione traccia un grafico a torta. Il parametro *handle* deve indicare un handle valido di stampante.

rec_x indica la coordinata x dell'angolo in alto a sinistra del rettangolo perimetrale.

rec_y indica la coordinata y dell'angolo in alto a sinistra del rettangolo perimetrale.

rec_x1 indica la coordinata x dell'angolo in basso a destra del rettangolo perimetrale.

rec_y1 indica la coordinata y dell'angolo in basso a destra del rettangolo perimetrale.

rad1_x indica la coordinata x del punto finale del primo raggio.

rad1_y indica la coordinata y del punto finale del primo raggio.

rad2_x indica la coordinata x del punto finale del secondo raggio.

rad2_y indica la coordinata y del punto finale del secondo raggio.

Esempio 1. Esempio di utilizzo di printer_draw_pie()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_pie($handle, 1, 1, 500, 500, 1, 1, 500, 1);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_rectangle (unknown)

Traccia un rettangolo

void **printer_draw_rectangle** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y) \linebreak

La funzione disegna un rettangolo.

Il parametro *handle* deve indicare un handle valido di stampante.

ul_x indica la coordinata x dell'angolo in alto a sinistra del rettangolo.

ul_y indica la coordinata y dell'angolo in alto a sinistra del rettangolo.

lr_x indica la coordinata x dell'angolo in basso a destra del rettangolo.

lr_y indica la coordinata y dell'angolo in basso a destra del rettangolo.

Esempio 1. Esempio di utilizzo di printer_draw_rectangle()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
```

```
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1, 1, 500, 500);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_roundrect (unknown)

Traccia un rettangolo con gli angoli arrotondati

void **printer_draw_roundrect** (resource handle, int ul_x, int ul_y, int lr_x, int lr_y, int width, int height) \linebreak

La funzione traccia un rettangolo con gli angoli arrotondati.

Il parametro *handle* deve indicare un handle valido di stampante.

ul_x indica la coordinata x dell'angolo in alto a sinistra del rettangolo.

ul_y indica la coordinata y dell'angolo in alto a sinistra del rettangolo.

lr_x indica la coordinata x dell'angolo in basso a destra del rettangolo.

lr_y indica la coordinata y dell'angolo in basso a destra del rettangolo.

width indica la larghezza dell'ellisse.

height indica l'altezza dell'ellisse.

Esempio 1. Esempio di utilizzo di printer_draw_roundrect()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "2222FF");
printer_select_brush($handle, $brush);

printer_draw_roundrect($handle, 1, 1, 500, 500, 200, 200);

printer_delete_brush($brush);
printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_draw_text (unknown)

Scrive un testo

void **printer_draw_text** (resource printer_handle, string testo, int x, int y) \linebreak

La funzione scrive il *testo* alla posizione *x*, *y* utilizzando il font selezionato. Il parametro *printer_handle* deve indicare un handle valido di stampante.

Esempio 1. Esempio di utilizzo di printer_draw_text()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial",72,48,400,false,false,false,0);
printer_select_font($handle, $font);
printer_draw_text($handle, "test", 10, 10);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_end_doc (unknown)

Chiude il documento

bool **printer_end_doc** (resource handle) \linebreak

Chiude il nuovo documento creato nello spooler della stampante. Ora il documento è pronto per essere stampato. Per un esempio vedere printer_start_doc(). Il parametro *handle* deve indicare un handle valido di stampante.

printer_end_page (unknown)

Chiude la pagina attiva

bool **printer_end_page** (resource handle) \linebreak

La funzione chiude la pagina attiva del documento corrente. Per un esempio vedere printer_start_doc(). Il parametro *handle* deve indicare un handle valido di stampante.

printer_get_option (unknown)

Recupera la configurazione della stampante

mixed **printer_get_option** (resource handle, string opzione) \linebreak

La funzione recupera il valore di configurazione per il parametro *opzione*. Il parametro *handle* deve indicare un handle valido di stampante. Vedere la funzione printer_set_option() per avere l'elenco dei parametri che sono disponibili; in aggiunta possono essere recuperate le informazioni sui parametri:

- *PRINTER_DEVICENAME* restituisce il nome della device della stampante.
- *PRINTER_DRIVERVERSION* restituisce la versione del driver della stampante.

Esempio 1. Esempio di utilizzo di printer_get_option()

```
$handle = printer_open();
print printer_get_option($handle, PRINTER_DRIVERVERSION);
printer_close($handle);
```

printer_list (unknown)

Restituisce un elenco delle stampanti collegate al server

array **printer_list** (int enumtype [, string nome [, int livello]]) \linebreak

La funzione elenca le stampanti disponibili e le loro capacità. Il parametro *livello* indica il livello delle informazioni richieste. I livelli possono essere 1,2,4 o 5. Il parametro *enumtype* deve essere valorizzato con una delle seguenti costanti:

- *PRINTER_ENUM_LOCAL*: elenca le stampanti installate localmente.
- *PRINTER_ENUM_NAME*: elenca le stampanti installate su *nome*, che può indicare un server, un dominio, un printer server.
- *PRINTER_ENUM_SHARED*: questo parametro non può essere utilizzato da solo, è necessario aggiungerlo in OR ad uno degli altri, ad esempio PRINTER_ENUM_LOCAL per rilevare le stampanti locali condivise.
- *PRINTER_ENUM_DEFAULT*: (solo Win9.x) elenca la stampante di default.
- *PRINTER_ENUM_CONNECTIONS*: (solo WinNT/2000) elenca le stampanti che l'utente può utilizzare.
- *PRINTER_ENUM_NETWORK*: (solo WinNT/2000) elenca le stampanti presenti nel dominio del computer. Opzione valida solo se *livello* è valorizzato a 1.
- *PRINTER_ENUM_REMOTE*: (solo WinNT/2000) elenca le stampanti di rete ed i printer server presenti nel dominio del computer. Opzione valida solo se *livello* è valorizzato a 1.

Esempio 1. Esempio di utilizzo di printer_list()

```
/* rileva le stampanti locali condivise */
var_dump( printer_list(PRINTER_ENUM_LOCAL | PRINTER_ENUM_SHARED) );
```

printer_logical_fontheight (unknown)

Restituisce l'altezza logica del font

int **printer_logical_fontheight** (resource handle, int altezza) \linebreak

La funzione calcola l'altezza logica del font per il parametro *altezza*. Il parametro *handle* deve indicare un handle valido di stampante.

Esempio 1. Esempio di utilizzo di printer_logical_fontheight()

```
$handle = printer_open();
print printer_logical_fontheight($handle, 72);
printer_close($handle);
```

printer_open (unknown)

Apre la connessione ad una stampante

mixed **printer_open** ([string NomePeriferica]) \linebreak

Questa funzione tenta di aprire una connessione con la stampante *NomePeriferica*, e restituisce un handle se ha successo oppure `FALSE` se fallisce.

Se non vengono passati parametri, la funzione tenta di aprire una connessione con la stampante di default (se non viene specificata in `php.ini` come `printer.default_printer`, il php tenta di determinarla).

Inoltre **printer_open()** attiva un device context.

Esempio 1. Esempio di utilizzo di printer_open()

```
$handle = printer_open("HP Deskjet 930c");
$handle = printer_open();
```

printer_select_brush (unknown)

Seleziona un pennello

void **printer_select_brush** (resource printer_handle, resource brush_handle) \linebreak

La funzione seleziona un pennello come oggetto attivo per il disegno nel device context corrente. Un pennello viene utilizzato per riempire le figure. Se si disegna un rettangolo il pennello viene utilizzato per disegnarne la forma, mentre si utilizza la penna per tracciarne i contorni. Se non si seleziona alcun pennello prima di disegnare delle figure, queste non saranno riempite. Il parametro *printer_handle* deve indicare un handle valido di stampante. Il parametro *brush_handle* deve indicare un handle valido di pennello.

Esempio 1. Esempio di utilizzo di printer_select_brush()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 2, "000000");
printer_select_pen($handle, $pen);
$brush = printer_create_brush(PRINTER_BRUSH_CUSTOM, "c:\\brush.bmp");
printer_select_brush($handle, $brush);

printer_draw_rectangle($handle, 1,1,500,500);

printer_delete_brush($brush);

$brush = printer_create_brush(PRINTER_BRUSH_SOLID, "000000");
printer_select_brush($handle, $brush);
printer_draw_rectangle($handle, 1,501,500,1001);
printer_delete_brush($brush);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_select_font (unknown)

Seleziona un font

void **printer_select_font** (resource printer_handle, resource font_handle) \linebreak

La funzione seleziona un font con cui scrivere testi. Il parametro *printer_handle* deve indicare un handle valido di stampante. Il parametro *font_handle* deve indicare un handle valido di font.

Esempio 1. Esempio di utilizzo di printer_select_font()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$font = printer_create_font("Arial", 148, 76, PRINTER_FW_MEDIUM, false, false, false, -
50);
printer_select_font($handle, $font);
printer_draw_text($handle, "PHP is simply cool", 40, 40);
printer_delete_font($font);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_select_pen (unknown)

Seleziona una penna

void **printer_select_pen** (resource printer_handle, resource pen_handle) \linebreak

La funzione seleziona una penna come oggetto attivo di disegno per il device context corrente. Si utilizza una per penna per tracciare linee e curve. Ad esempio se si deve tracciare un linea singola si usa una penna. Se si disegna un rettangolo, si utilizza la penna per tracciare i bordi, mentre con il pennello si riempie l'interno. Se non si seleziona una penna prima di tracciare delle figure, queste non avranno i bordi. Il parametro *printer_handle* deve indicare un handle valido di stampante. Il parametro *pen_handle* deve indicare un handle valido di penna.

Esempio 1. Esempio di utilizzo di printer_select_pen()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

$pen = printer_create_pen(PRINTER_PEN_SOLID, 30, "2222FF");
printer_select_pen($handle, $pen);

printer_draw_line($handle, 1, 60, 500, 60);

printer_delete_pen($pen);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_set_option (unknown)

Configura la connessione con la stampante

bool **printer_set_option** (resource handle, int opzione, mixed valore) \linebreak

La funzione valorizza le seguenti opzioni per la connessione corrente. Il parametro *handle* deve indicare un handle valido di stampante. Per il parametro *opzione* si può utilizzare una delle seguenti costanti:

- *PRINTER_COPIES*: indica quante copie si debbano stampare, *valore* deve essere un intero.
- *PRINTER_MODE*: specifica il tipo di dati (text, raw or emf), *valore* deve essere una stringa.
- *PRINTER_TITLE*: specifica il nome del documento, *valore* deve essere una stringa.
- *PRINTER_ORIENTATION*: specifica l'orientamento del foglio, *valore* può essere o PRINTER_ORIENTATION_PORTRAIT o PRINTER_ORIENTATION_LANDSCAPE
- *PRINTER_RESOLUTION_Y*: specifica la risoluzione y in DPI, *valore* deve essere un intero.
- *PRINTER_RESOLUTION_X*: specifica la risoluzione x in DPI, *valore* deve essere un intero.
- *PRINTER_PAPER_FORMAT*: specifica il formato predefinito della carta, impostare *valore* a PRINTER_FORMAT_CUSTOM se si vuole impostare un formato personalizzato con PRINTER_PAPER_WIDTH e PRINTER_PAPER_LENGTH. Il parametro *valore* può essere una delle seguenti costanti.
  - *PRINTER_FORMAT_CUSTOM*: specifica un formato personalizzato.
  - *PRINTER_FORMAT_LETTER*: specifica il formato letter (8 1/2- per 11-pollici).
  - *PRINTER_FORMAT_LETTER*: specifica il formato legal (8 1/2- per 14-pollici).
  - *PRINTER_FORMAT_A3*: specifica il formato A3 (297- per 420-millimetri).
  - *PRINTER_FORMAT_A4*: specifica il formato A4 (210- per 297-millimetri).
  - *PRINTER_FORMAT_A5*: specifica il formato A5 (148- per 210-millimetri).
  - *PRINTER_FORMAT_B4*: specifica il formato B4 (250- per 354-millimetri).
  - *PRINTER_FORMAT_B5*: specifica il formato B5 (182- per 257-millimetri).
  - *PRINTER_FORMAT_FOLIO*: specifica il formato FOLIO (8 1/2- per 13-pollici).
- *PRINTER_PAPER_LENGTH*: se PRINTER_PAPER_FORMAT è impostato a PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_LENGTH specifica la lunghezza personalizzata in mm, *valore* deve essere un intero.
- *PRINTER_PAPER_WIDTH*: se PRINTER_PAPER_FORMAT è impostato a PRINTER_FORMAT_CUSTOM, PRINTER_PAPER_WIDTH specifica la larghezza personalizzata in mm, *valore* deve essere un intero.
- *PRINTER_SCALE*: specifica il fattore per il quale l'output della stampante deve essere dimensionato. La dimensione dalle pagine viene modificata dalla dimensione fisica di un fattore pari a scala/100. Ad esempio se simposta scala a 50, l'output sarà la metà della dimensione originale. *Valore* deve essere un intero.
- *PRINTER_BACKGROUND_COLOR*: specifica il colore di background per il device context corrente, *valore* deve essere una stringa contenente il colore in formato RGB esadecimale, ad esempio "005533".

- *PRINTER_TEXT_COLOR*: specifica il colore del testo per il device context corrente, *valore* deve essere una stringa contenente il colore in formato RGB esadecimale, ad esempio "005533".
- *PRINTER_TEXT_ALIGN*: specifica l'allineamento del testo per il device context corrente, *valore* può essere la combinazione tramite OR delle seguenti costanti:
  - *PRINTER_TA_BASELINE*: il testo sarà allineato alla linea base.
  - *PRINTER_TA_BOTTOM*: il testo sarà allineato in basso.
  - *PRINTER_TA_TOP*: il testo sarà allineato in alto.
  - *PRINTER_TA_CENTER*: il testo sarà centrato.
  - *PRINTER_TA_LEFT*: il testo sarà allineato a sinistra.
  - *PRINTER_TA_RIGHT*: il testo sarà allineato a destra.

Esempio 1. Esempio di utilizzo di printer_set_option()

```
$handle = printer_open();
printer_set_option($handle, PRINTER_SCALE, 75);
printer_set_option($handle, PRINTER_TEXT_ALIGN, PRINTER_TA_LEFT);
printer_close($handle);
```

printer_start_doc (unknown)

Inizia un nuovo documento

bool **printer_start_doc** (resource handle [, string documento]) \linebreak

La funzione crea un nuovo documento nello spooler della stampante. Il documento può contenere diverse pagine, la funzione è utilizzata per inserire un job di stampa nello spooler. Il parametro *handle* deve essere un handle valido di una stampante. Il parametro opzionale *documento* può esesre usato per indicare un nome alternativo al documento.

Esempio 1. Esempio di utilizzo di printer_start_doc()

```
$handle = printer_open();
printer_start_doc($handle, "My Document");
printer_start_page($handle);

printer_end_page($handle);
printer_end_doc($handle);
printer_close($handle);
```

printer_start_page (unknown)

Inizia una nuova pagina

bool **printer_start_page** (resource handle) \linebreak

La funzione crea una nuova pagina nel documento attivo. Per un esempio vedere printer_start_doc(). Il parametro *handle* deve essere un handle valido di stampante.

printer_write (unknown)

Scrive dei dati sulla stampante

bool **printer_write** (resource handle, string contenuto) \linebreak

La funzione scrive *contenuto* direttamente alla stampante, e restituisce TRUE se ha successo oppure FALSE se non riesce.

Il parametro *handle* deve essere un handle valido di una stampante.

Esempio 1. Esempio di utilizzo di printer_write()

```
$handle = printer_open();
printer_write($handle, "Testo da scrivere");
printer_close($handle);
```

LXXXIII. Pspell Functions

These functions allow you to check the spelling of a word and offer suggestions.

You need the aspell and pspell libraries, available from http://aspell.sourceforge.net/ and http://pspell.sourceforge.net/ respectively, and add the `--with-pspell[=dir]` option when compiling php.

pspell_add_to_personal (PHP 4 >= 4.0.2)

Add the word to a personal wordlist

int **pspell_add_to_personal** (int dictionary_link, string word) \linebreak

pspell_add_to_personal() adds a word to the personal wordlist. If you used pspell_new_config() with pspell_config_personal() to open the dictionary, you can save the wordlist later with pspell_save_wordlist(). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Esempio 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_add_to_session (PHP 4 >= 4.0.2)

Add the word to the wordlist in the current session

int **pspell_add_to_session** (int dictionary_link, string word) \linebreak

pspell_add_to_session() adds a word to the wordlist associated with the current session. It is very similar to pspell_add_to_personal()

pspell_check (PHP 4 >= 4.0.2)

Check a word

bool **pspell_check** (int dictionary_link, string word) \linebreak

pspell_check() checks the spelling of a word and returns `TRUE` if the spelling is correct, `FALSE` if not.

Esempio 1. pspell_check()

```
$pspell_link = pspell_new ("en");

if (pspell_check ($pspell_link, "testt")) {
    echo "This is a valid spelling";
```

```
} else {
    echo "Sorry, wrong spelling";
}
```

pspell_clear_session (PHP 4 >= 4.0.2)

Clear the current session

int **pspell_clear_session** (int dictionary_link) \linebreak

pspell_clear_session() clears the current session. The current wordlist becomes blank, and, for example, if you try to save it with pspell_save_wordlist(), nothing happens.

Esempio 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_clear_session ($pspell_link);
pspell_save_wordlist ($pspell_link);    //"Vlad" will not be saved
```

pspell_config_create (PHP 4 >= 4.0.2)

Create a config used to open a dictionary

int **pspell_config_create** (string language [, string spelling [, string jargon [, string encoding]]]) \linebreak

pspell_config_create() has a very similar syntax to pspell_new(). In fact, using **pspell_config_create()** immediatelly followed by pspell_new_config() will produce the exact same result. However, after creating a new config, you can also use **pspell_config_*()** functions before calling pspell_new_config() to take advantage of some advanced functionality.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are ’american’, ’british’, and ’canadian’.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are ’utf-8’, ’iso8859-*’, ’koi8-r’, ’viscii’, ’cp1252’, ’machine unsigned 16’, ’machine unsigned 32’. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Esempio 1. pspell_config_create()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_personal ($pspell_config, "en");
```

pspell_config_ignore (PHP 4 >= 4.0.2)

Ignore words less than N characters long

int **pspell_config_ignore** (int dictionary_link, int n) \linebreak

pspell_config_ignore() should be used on a config before calling pspell_new_config(). This function allows short words to be skipped by the spellchecker. Words less then n characters will be skipped.

Esempio 1. pspell_config_ignore()

```
$pspell_config = pspell_config_create ("en");
pspell_config_ignore($pspell_config, 5);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "abcd");    //will not result in an error
```

pspell_config_mode (PHP 4 >= 4.0.2)

Change the mode number of suggestions returned

int **pspell_config_mode** (int dictionary_link, int mode) \linebreak

pspell_config_mode() should be used on a config before calling pspell_new_config(). This function determines how many suggestions will be returned by pspell_suggest().

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)

Esempio 1. pspell_config_mode()

```
$pspell_config = pspell_config_create ("en");
pspell_config_mode($pspell_config, PSPELL_FAST);
$pspell_link = pspell_new_config($pspell_config);
pspell_check($pspell_link, "thecat");
```

pspell_config_personal (PHP 4 >= 4.0.2)

Set a file that contains personal wordlist

int **pspell_config_personal** (int dictionary_link, string file) \linebreak

pspell_config_personal() should be used on a config before calling pspell_new_config(). The personal wordlist will be loaded and used in addition to the standard one after you call pspell_new_config(). If the file does not exist, it will be created. The file is also the file where pspell_save_wordlist() will save personal wordlist to. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Esempio 1. pspell_config_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_repl (PHP 4 >= 4.0.2)

Set a file that contains replacement pairs

int **pspell_config_repl** (int dictionary_link, string file) \linebreak

pspell_config_repl() should be used on a config before calling pspell_new_config(). The replacement pairs improve the quality of the spellchecker. When a word is misspelled, and a proper suggestion was not found in the list, pspell_store_replacement() can be used to store a replacement pair and then pspell_save_wordlist() to save the wordlist along with the replacement pairs. The file should be writable by whoever php runs as (e.g. nobody). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Esempio 1. pspell_config_repl()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_runtogether (PHP 4 >= 4.0.2)

Consider run-together words as valid compounds

int **pspell_config_runtogether** (int dictionary_link, bool flag) \linebreak

pspell_config_runtogether() should be used on a config before calling pspell_new_config(). This function determines whether run-together words will be treated as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Esempio 1. pspell_config_runtogether()

```
$pspell_config = pspell_config_create ("en");
pspell_config_runtogether ($pspell_config, true);
$pspell_link = pspell_new_config ($pspell_config);
pspell_check ($pspell_link, "thecat");
```

pspell_config_save_repl (PHP 4 >= 4.0.2)

Determine whether to save a replacement pairs list along with the wordlist

int **pspell_config_save_repl** (int dictionary_link, bool flag) \linebreak

pspell_config_save_repl() should be used on a config before calling pspell_new_config(). It determines whether pspell_save_wordlist() will save the replacement pairs along with the wordlist. Usually there is no need to use this function because if pspell_config_repl() is used, the replacement pairs will be saved by pspell_save_wordlist() anyway, and if it is not, the replacement pairs will not be saved. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

pspell_new (PHP 4 >= 4.0.2)

Load a new dictionary

int **pspell_new** (string language [, string spelling [, string jargon [, string encoding [, int mode]]]]) \linebreak

pspell_new() opens up a new dictionary and returns the dictionary link identifier for use in other pspell functions.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)
- `PSPELL_RUN_TOGETHER` - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, `PSPELL_FAST`, `PSPELL_NORMAL` and `PSPELL_BAD_SPELLERS` are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Esempio 1. pspell_new()

```
$pspell_link = pspell_new ("en", "", "", "",
                           (PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_new_config (PHP 4 >= 4.0.2)

Load a new dictionary with settings based on a given config

int **pspell_new_config** (int config) \linebreak

pspell_new_config() opens up a new dictionary with settings specified in a config, created with with pspell_config_create() and modified with **pspell_config_*()** functions. This method provides you with the most flexibility and has all the functionality provided by pspell_new() and pspell_new_personal().

The config parameter is the one returned by pspell_config_create() when the config was created.

Esempio 1. pspell_new_config()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);
```

pspell_new_personal (PHP 4 >= 4.0.2)

Load a new dictionary with personal wordlist

int **pspell_new_personal** (string personal, string language [, string spelling [, string jargon [, string encoding [, int mode]]]]) \linebreak

pspell_new_personal() opens up a new dictionary with a personal wordlist and returns the dictionary link identifier for use in other pspell functions. The wordlist can be modified and saved with pspell_save_wordlist(), if desired. However, the replacement pairs are not saved. In order to save replacement pairs, you should create a config using pspell_config_create(), set the personal

wordlist file with pspell_config_personal(), set the file for replacement pairs with pspell_config_repl(), and open a new dictionary with pspell_new_config().

The personal parameter specifies the file where words added to the personal list will be stored. It should be an absolute filename beginning with '/' because otherwise it will be relative to $HOME, which is "/root" for most systems, and is probably not what you want.

The language parameter is the language code which consists of the two letter ISO 639 language code and an optional two letter ISO 3166 country code after a dash or underscore.

The spelling parameter is the requested spelling for languages with more than one spelling such as English. Known values are 'american', 'british', and 'canadian'.

The jargon parameter contains extra information to distinguish two different words lists that have the same language and spelling parameters.

The encoding parameter is the encoding that words are expected to be in. Valid values are 'utf-8', 'iso8859-*', 'koi8-r', 'viscii', 'cp1252', 'machine unsigned 16', 'machine unsigned 32'. This parameter is largely untested, so be careful when using.

The mode parameter is the mode in which spellchecker will work. There are several modes available:

- `PSPELL_FAST` - Fast mode (least number of suggestions)
- `PSPELL_NORMAL` - Normal mode (more suggestions)
- `PSPELL_BAD_SPELLERS` - Slow mode (a lot of suggestions)
- `PSPELL_RUN_TOGETHER` - Consider run-together words as legal compounds. That is, "thecat" will be a legal compound, athough there should be a space between the two words. Changing this setting only affects the results returned by pspell_check(); pspell_suggest() will still return suggestions.

Mode is a bitmask constructed from different constants listed above. However, `PSPELL_FAST`, `PSPELL_NORMAL` and `PSPELL_BAD_SPELLERS` are mutually exclusive, so you should select only one of them.

For more information and examples, check out inline manual pspell website:http://pspell.sourceforge.net/.

Esempio 1. pspell_new_personal()

```
$pspell_link = pspell_new_personal ("/var/dictionaries/custom.pws",
        "en", "", "", "", PSPELL_FAST|PSPELL_RUN_TOGETHER));
```

pspell_save_wordlist (PHP 4 >= 4.0.2)

Save the personal wordlist to a file

int **pspell_save_wordlist** (int dictionary_link) \linebreak

pspell_save_wordlist() saves the personal wordlist from the current session. The dictionary has to be open with pspell_new_personal(), and the location of files to be saved specified with pspell_config_personal() and (optionally) pspell_config_repl(). Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Esempio 1. pspell_add_to_personal()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/tmp/dicts/newdict");
$pspell_link = pspell_new_config ($pspell_config);

pspell_add_to_personal ($pspell_link, "Vlad");
pspell_save_wordlist ($pspell_link);
```

pspell_store_replacement (PHP 4 >= 4.0.2)

Store a replacement pair for a word

int **pspell_store_replacement** (int dictionary_link, string misspelled, string correct) \linebreak

pspell_store_replacement() stores a replacement pair for a word, so that replacement can be returned by pspell_suggest() later. In order to be able to take advantage of this function, you have to use pspell_new_personal() to open the dictionary. In order to permanently save the replacement pair, you have to use pspell_config_personal() and pspell_config_repl() to set the path where to save your custom wordlists, and then use pspell_save_wordlist() for the changes to be written to disk. Please, note that this function will not work unless you have pspell .11.2 and aspell .32.5 or later.

Esempio 1. pspell_store_replacement()

```
$pspell_config = pspell_config_create ("en");
pspell_config_personal ($pspell_config, "/var/dictionaries/custom.pws");
pspell_config_repl ($pspell_config, "/var/dictionaries/custom.repl");
$pspell_link = pspell_new_config ($pspell_config);

pspell_store_replacement ($pspell_link, $misspelled, $correct);
pspell_save_wordlist ($pspell_link);
```

pspell_suggest (PHP 4 >= 4.0.2)

Suggest spellings of a word

array **pspell_suggest** (int dictionary_link, string word) \linebreak

pspell_suggest() returns an array of possible spellings for the given word.

Esempio 1. pspell_suggest()

```
$pspell_link = pspell_new ("en");

if (!pspell_check ($pspell_link, "testt")) {
    $suggestions = pspell_suggest ($pspell_link, "testt");

    foreach ($suggestions as $suggestion) {
        echo "Possible spelling: $suggestion<br>";
    }
}
```

LXXXIV. GNU Readline

The readline() functions implement an interface to the GNU Readline library. These are functions that provide editable command lines. An example being the way Bash allows you to use the arrow keys to insert characters or scroll through command history. Because of the interactive nature of this library, it will be of little use for writing Web applications, but may be useful when writing scripts meant to be run from a shell.

Requirements

To use the readline functions, you need to install libreadline and compile PHP with support for readline.

Installation

To compile PHP with readline support, you need to configure PHP `--with-readline` after you've installed libreadline. You can find libreadline on the home page of the GNU Readline project, at http://cnswww.cns.cwru.edu/~chet/readline/rltop.html. It's maintained by Chet Ramey, who's also the author of Bash.

Runtime Configuration

Questa estensione non definisce alcuna direttiva di configurazione

Resource types

Questa estensione non definisce alcun tipo di risorsa.

Predefined constants

Questa estensione non definisce alcuna costante.

readline (PHP 4 >= 4.0.0)

Reads a line

string **readline** ([string prompt]) \linebreak

This function returns a single string from the user. You may specify a string with which to prompt the user. The line returned has the ending newline removed. You must add this line to the history yourself using readline_add_history().

Esempio 1. readline()

```
//get 3 commands from user
for ($i=0; $i < 3; $i++) {
        $line = readline ("Command: ");
        readline_add_history ($line);
}

//dump history
print_r (readline_list_history());

//dump variables
print_r (readline_info());
```

readline_add_history (PHP 4 >= 4.0.0)

Adds a line to the history

void **readline_add_history** (string line) \linebreak

This function adds a line to the command line history.

readline_clear_history (PHP 4 >= 4.0.0)

Clears the history

bool **readline_clear_history** (void) \linebreak

This function clears the entire command line history.

readline_completion_function (PHP 4 >= 4.0.0)

Registers a completion function

bool **readline_completion_function** (string line) \linebreak

This function registers a completion function. You must supply the name of an existing function which accepts a partial command line and returns an array of possible matches. This is the same kind of functionality you'd get if you hit your tab key while using Bash.

readline_info (PHP 4 >= 4.0.0)

Gets/sets various internal readline variables

mixed **readline_info** ([string varname [, string newvalue]]) \linebreak

If called with no parameters, this function returns an array of values for all the setting readline uses. The elements will be indexed by the following values: done, end, erase_empty_line, library_version, line_buffer, mark, pending_input, point, prompt, readline_name, and terminal_name.

If called with one parameter, the value of that setting is returned. If called with two parameters, the setting will be changed to the given value.

readline_list_history (PHP 4 >= 4.0.0)

Lists the history

array **readline_list_history** (void) \linebreak

This function returns an array of the entire command line history. The elements are indexed by integers starting at zero.

readline_read_history (PHP 4 >= 4.0.0)

Reads the history

bool **readline_read_history** (string filename) \linebreak

This function reads a command history from a file.

readline_write_history (PHP 4 >= 4.0.0)

Writes the history

bool **readline_write_history** (string filename) \linebreak

This function writes the command history to a file.

LXXXV. GNU Recode functions

This module contains an interface to the GNU Recode library, version 3.5. To be able to use the functions defined in this module you must compile you PHP interpreter using the --with-recode option. In order to do so, you must have GNU Recode 3.5 or higher installed on your system.

The GNU Recode library converts files between various coded character sets and surface encodings. When this cannot be achieved exactly, it may get rid of the offending characters or fall back on approximations. The library recognises or produces nearly 150 different character sets and is able to convert files between almost any pair. Most RFC 1345 character sets are supported.

recode (PHP 4 >= 4.0.0)

Recode a string according to a recode request

string **recode** (string request, string string) \linebreak

Nota: This is an alias for recode_string(). It has been added in PHP 4.

recode_file (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Recode from file to file according to recode request

bool **recode_file** (string request, resource input, resource output) \linebreak

Recode the file referenced by file handle *input* into the file referenced by file handle *output* according to the recode *request*. Returns FALSE, if unable to comply, TRUE otherwise.

This function does not currently process filehandles referencing remote files (URLs). Both filehandles must refer to local files.

Esempio 1. Basic recode_file() example

```
$input = fopen ('input.txt', 'r');
$output = fopen ('output.txt', 'w');
recode_file ("us..flat", $input, $output);
```

recode_string (PHP 3>= 3.0.13, PHP 4 >= 4.0.0)

Recode a string according to a recode request

string **recode_string** (string request, string string) \linebreak

Recode the string *string* according to the recode request *request*. Returns the recoded string or FALSE, if unable to perform the recode request.

A simple recode request may be "lat1..iso646-de". See also the GNU Recode documentation of your installation for detailed instructions about recode requests.

Esempio 1. Basic recode_string() example:

```
print recode_string ("us..flat", "The following character has a diacritical mark: &aacute
```

LXXXVI. Funzioni per le espressioni regolari (Perl compatibili)

La sintassi utilizzata dalle espressioni regolari di queste funzioni ricorda da vicino Perl. Le espressioni regolari devono essere racchiuse tra delimitatori, ad esempio /. Ogni carattere che non sia alfanumerico od il backslash (\) può essere usato come delimitatore. Se il carattere usato come delimitatore deve essere utilizzato all'interno dell'espressione, questo deve essere preceduto dal carattere di escape (\). Infine, a partire dalla versione 4.0.4 di PHP, si possono anche usare i delimitatori stile Perl come (), {}, [], ed <>.

Il delimitatore finale può essere seguito da vari modificatori che agiscono sul criterio di riconoscimento. Per maggiori informazioni si rimanda al capitolo Modificatori di criterio (Pattern Modifiers).

Utilizzando le funzioni POSIX-esteso il PHP è in grado di supportare le espressioni regolari scritte con la sintassi POSIX-esteso.

Requisiti

Il supporto delle espressioni regolari è ottenuto mediante la libreria PCRE, che è un software open source, scritto da Philip Hazel, ed il cui copyright è detenuto dalla Università di Cambridge, Inghilterra. La libreria è disponibile all'indirizzo ftp://ftp.csx.cam.ac.uk/pub/software/programming/pcre/.

Installazione

A partire dalla versione 4.2.0 di PHP queste funzioni sono abilitate per default. Invece, nelle vecchie versioni, occorre configurare e compilare il PHP con `--with-pcre-regex[=DIR]` per attivare queste funzioni. Viceversa la funzioni PCRE possono essere disabilitate utilizzando `--without-pcre-regex`.

Configurazioni per l'esecuzione (Runtime Configuration)

Questa estensione non definisce alcuna direttiva di configurazione

Tipologia di risorse

Questa estensione non definisce alcun tipo di risorsa.

Costanti pre-definite

`PREG_PATTERN_ORDER` `PREG_SET_ORDER` `PREG_SPLIT_NO_EMPTY`

PREG_SPLIT_DELIM_CAPTURE

Esempi

Esempio 1. Esempi di espressioni di riconoscimento valide

- /<\/\w+>/
- |(\d{3})-\d+|Sm
- /^(?i)php[34]/
- {^\s+(\s+)?$}

Esempio 2. Esempi di espressioni errate

- /href='(.*)' - manca il delimitatore finale
- /\w+\s*\w+/J - il modificatore 'J' è sconosciuto
- 1-\d3-\d3-\d4| - manca il delimitatore iniziale

Modificatori di criterio (Pattern Modifiers) (unknown)

Questo capitolo descrive i possibili modificatori applicabili ad una espressione regolare

Di seguito saranno elencati i valori attualmente previsti. I nomi posti tra parentesi si riferiscono ai corrispettivi nomi usati internamente da PCRE.

i (PCRE_CASELESS)

Se si attiva questo modificatore, l'espressione regolare viene riconosciuta senza distinguere tra le lettere maiuscole e minuscole.

m (PCRE_MULTILINE)

Per default, PCRE tratta il testo su cui fare la ricerca come una "singola linea" di caratteri (anche se in realtà può contenere diversi "a capo" (newline)). Il carattere di "inizio riga" (^) indica solamente l'inizio del testo passato. Analogamente il carattere di "fine riga" ($) indica la fine del testo o prima se vi sono dei caratteri di "a capo" (a meno che non sia attivato il modificatore *D*). In questo modo si comporta anche Perl.

Invece quando viene indicato questo modificatore, "inizio riga" e "fine riga" vengono identificati in base ai caratteri di "a capo" presenti nel testo (rispettivamente subito dopo e subito prima di questo carattere). Questo comportamento è equivalente al modificatore /m di Perl. Se nel testo passato non vi sono caratteri di "a capo" o non vi sono occorrenze dei caratteri ^ o $ nell'espressione regolare, l'uso di questo modificatore non ha effetto.

s (PCRE_DOTALL)

Se si attiva questo modificatore, il carattere . usato nell'espressione regolare indica tutti i possibili caratteri compreso il carattere di "a capo" (newline). Senza questo modificatore il carattere "a capo" viene escluso. Questo modificare è equivalente a /s di Perl. Una espressione regolare contenente una forma negativa, come [^a], riconosce sempre un "a capo" a prescindere da questo modificatore.

x (PCRE_EXTENDED)

Se si attiva questo modificatore, verranno ignorati gli spazi presenti nell'espressione regolare, tranne quelli preceduti dal carattere di escape (\) o posti all'interno di una classe di caratteri. Saranno, inoltre, ignorati i caratteri posti tra il simbolo # (se non è preceduto dall'escape, ed è posto al di fuori di una classe di caratteri) ed il successivo carattere di "a capo" (newline). Questo comportamento equivale al flag /x di Perl e consente di inserire dei commenti all'interno di espressioni regolari complesse. Gli spazi bianchi non devono mai comparire all'interno di sequenze speciali di caratteri, come, ad esempio, la sequenza (?(che introduce ad un criterio condizionale.

e

Se viene specificato questo modificatore, la funzione preg_replace() valuta la stringa di sostituzione come codice PHP, quindi utilizza il risultato come testo da sostituire alle stringhe cercate.

Soltanto preg_replace() utilizza questo modificatore; le altre funzioni di PCRE lo ignorano.

A (PCRE_ANCHORED)

Se si specifica questo modificatore, si forza un 'ancoraggio' del criterio di ricerca. In pratica questo viene costretto a riconoscere il testo su cui si fa la ricerca solo dall'inizio. Questo effetto può essere ottenuto anche con particolari costruzioni dell'espressione regolare, che rappresentano gli unici modi utilizzabili in Perl per ottenere il medesimo scopo.

D (PCRE_DOLLAR_ENDONLY)

L'uso di questo modificatore forza il carattere $ dell'espressione regolare a indicare la fine della stringa oggetto della ricerca. Senza questo modificatore il carattere $ indica la posizione subito prima dell'ultimo carattere se questo è un "a capo" (ma comunque non prima di ogni altro "a capo"). Questo modificatore viene ignorato se è attivato il modificatore *m*. Non vi sono flag equivalenti in Perl.

S

Quando una espressione regolare è destinata ad essere utilizzata diverse volte, vale la pena dedicare del tempo ad ottimizzare la velocità di riconoscimento. L'uso di questo modificatore permette questa analisi. Al momento lo studio della velocità è significativo per i criteri di ricerca "non ancorati", cioè espressioni che non hanno un carattere di partenza fisso.

U (PCRE_UNGREEDY)

Questo modificatore inverte la "voracità" delle occorrenze, in modo da non essere voraci per default, ma lo tornano ad essere se seguiti da "?". Questo flag non è compatibile con Perl. Questo comportamento può anche essere settato dalla sequenza (?U) posta all'interno dell'espressione regolare.

X (PCRE_EXTRA)

Questo modificatore attiva funzionalità addizionali di PCRE che sono incompatibili con Perl. Ogni backslash (\) posto nell'espressione regolare che non sia seguito da una lettera con significato speciale causa un errore, ciò per riservare queste sequenze a future espansioni. Per default, Perl considera il backslash (\) seguito da una lettera priva di significato speciale come un qualsiasi testo. Al momento non vi sono altre caratteristiche gestite tramite questo modificatore.

u (PCRE_UTF8)

Questo modificatore attiva funzionalità di PCRE che sono incompatibili con Perl. Le stringhe di ricerca sono considerate come UTF-8. Questo modificatore è disponibile dalla versione 4.1.0 di PHP.

Sintassi delle espressioni regolari (unknown)

Descrizione della sintassi utilizzata da PCRE per la definizione delle espressioni regolari

La libreria PCRE è costituita da una serie di funzioni che utilizzano come criterio di riconoscimento le espressioni regolari mutuando la stessa sintassi e la stessa semantica di Perl 5, a parte qualche lieve differenza descritta di seguito. L'attuale implementazione corrisponde alla versione 5.005 di Perl.

In questo capitolo saranno descritte le differenze rispetto a Perl 5.005.

1. Per default, lo spazio bianco indica tutti i caratteri riconoscibili dalla funzione isspace() della libreria C, ciò non pregiudica la possibilità di compilare PCRE con un set di caratteri alternativi. Normalmente isspace() riconosce gli spazi, il salto pagina, il carattere 'a capo' (newline), il carattere carriage return, la tabulazione orizzontale e verticale. Perl 5 non riconosce nel set dei caratteri indicati dallo "spazio bianco" la tabulazione verticale. Infatti il carattere di escape \v è stato presente per diverso tempo nella documentazione di Perl, ma di fatto non viene riconosciuto. Tuttavia lo stesso carattere viene trattato come spazio bianco fino alla versione 5.002 di Perl. Nelle versioni 5.004 e 5.005 non viene riconosciuto il carattere \s.
2. PCRE non supporta occorrenze ripetute in espressioni che "guardano avanti". Perl lo permette, ma non nel modo a cui si possa pensare. Ad esempio, (?!a){3}, non indica che i tre caratteri

successivi non debbano essere delle "a", ma indica per tre volte che il carattere successivo non debba essere una "a".

3. Il riconoscimento di criteri parziali che possono verificarsi all'interno di espressioni che "guardano avanti" negative sono contati, ma i loro riferimenti non sono inseriti nel vettore degli offset. Perl valorizza le sue variabili da qualsiasi criterio che sia riconosciuto prima che l'espressione regolare fallisca nel riconoscere qualcosa, ma questo solo se l'espressione che "guarda avanti" contiene almeno un ramo alternativo.
4. Sebbene il carattere di 0 binario sia supportato nella stringa in cui si deve svolgere la ricerca, non è ammesso nel testo dell'espressione regolare, questo perché il testo viene passato come stringa C conclusa dal carattere zero. Si può comunque utilizzare la sequenza "\0" per richiedere il riconoscimento dello zero binario.
5. Non sono supportate le seguenti sequenze di escape di Perl: \l, \u, \L, \U, \E e \Q. Infatti queste sono implementate nelle funzioni di gestione delle stringhe di Perl e non nel suo motore di riconoscimento delle espressioni regolari.
6. L'asserzione di Perl \G non è supportata.
7. Ovviamente PCRE non supporta il costrutto (?{code}).
8. Al momento in cui si scrive, in Perl 5.005_02 vi sono alcune particolarità riguardanti la parametrizzazione delle stringhe catturate quando parte del criterio di riconoscimento viene ripetuto. Ad esempio, il riconoscimento di "aba" con il criterio /^(a(b)?)+$/, valorizza $2 con la lettera "b", usando il testo "aabbaa" con il criterio /^(aa(bb)?)+$/ non si ha la valorizzazione di $2. Tuttavia se il criterio di riconoscimento viene variato in /^(aa(b(b))?)+$/, sia $2 che $3 vengono valorizzate. Nelle versione 5.004 di Perl, la variabile $2 viene valorizzata in entrambi i casi, come pure è `TRUE` in PCRE. Se in futuro Perl cambierà nella gestione anche PCRE potrà cambiare di conseguenza.
9. Un'altra discrepanza non ancora risolta riguarda il criterio di riconoscimento /^(a)?(?(1)a|b)+$/, che in Perl 5.005_02 riconosce la stringa "a", mentre non accade in PCRE. Tuttavia in entrambi (Perl e PCRE) il riconoscimento di "a" con il criterio /^(a)?a/ non valorizza $1.
10. PCRE prevede alcune estensione alla gestione delle espressioni regolari di Perl:
    a. Sebbene le asserzioni che "guardano indietro" richiedano testi di lunghezza fissa, è ammesso che ciascun ramo alternativo del criterio di ricerca possa intercettare stringhe di lunghezza differente. Al contrario Perl 5.005 richiede che tutte le stringhe abbiamo la stessa lunghezza.
    b. Se viene settato PCRE_DOLLAR_ENDONLY e non viene attivato PCRE_MULTILINE, il meta-carattere $ indica soltanto la fine della stringa.
    c. Se si attiva PCRE_EXTRA, un carattere backslash (\) seguito da una lettera che non abbia un significato speciale genera un errore.
    d. Se si attiva PCRE_UNGREEDY, si inverte la "voracità" delle occorrenze, in modo da non essere voraci per default, ma lo tornano ad essere se seguiti da "?".

Introduzione

Di seguito verrà descritta la sintassi e la semantica delle espressioni regolari così come sono supportate da PCRE. La descrizione delle espressioni regolari può essere reperita anche nei manuali di Perl e in numerosi altri libri, alcuni dei quali ricchi di esempi. Ad esempio il libro di Jeffrey Friedl

intitolato "Mastering Regular Expressions", edito da O'Reilly (ISBN 1-56592-257-3), li tratta in modo dettagliato. Questa descrizione, invece, viene intesa come una documentazione di riferimento. Una espressione regolare è un criterio utilizzato per identificare parti di un testo partendo da sinistra verso destra. La maggior parte delle lettere identifica se stessa nel testo oggetto del riconoscimento. Ad esempio il banale testo `La volpe veloce` intercetta la parte del testo uguale all'espressione regolare.

Meta-caratteri

La potenza delle espressioni regolari deriva dalla possibilità di inserire criteri alternativi oppure ripetuti. Questi sono codificati nel criterio di ricerca tramite l'uso di *meta-caratteri*, lettere che non indicano se stesse, ma sono interpretate in modo particolare.

Esistono due differenti set di meta-caratteri: quelli che sono riconosciuti ovunque tranne che all'interno di parentesi quadrate, e quelli che sono riconosciuti all'interno di parentesi quadrate. I meta-caratteri che si usano all'esterno delle parentesi quadrate sono:

\

carattere di escape generico con diversi utilizzi

^

indica l'inizio del testo (o della linea in modalità multi-linea)

$

indica la fine del testo (o della linea in modalità multi-linea)

.

indica qualsiasi carattere tranne "a capo" (per default)

[

carattere di inizio della definizione di classe

]

carattere di fine della definizione di classe

|

inizio di un ramo alternativo

(

inizio di un criterio di riconoscimento parziale

)

fine del criterio di riconoscimento parziale

?

estende il significato di (, oppure 0 o 1 occorrenza, oppure occorrenza minima

*

0 o più occorrenze

+

1 o più occorrenze

{

inizia l’intervallo minimo/massimo di occorrenze

}

termina l’intervallo minimo/massimo di occorrenze

La parte del criterio che si trova tra parentesi quadrate viene detta "classe di caratteri". In una classe di caratteri i meta-caratteri previsti sono:

**

carattere di escape generico con diversi utilizzi

^

nega la classe, ma solo se posto all’inizio

-

indica un intervallo

]

chiude la classe di caratteri

Le sezioni seguenti descriveranno l’uso di ciascuno dei meta-caratteri.

Backslash

Il carattere backslash (\) ha diversi utilizzi. Primo uso: se viene anteposto a caratteri non alfanumerici, rimuove gli eventuali significati speciali che il carattere può avere. Questo utilizzo di backslash come carattere di escape può essere svolto sia all’interno delle classi di caratteri sia all’esterno.

Ad esempio, un criterio che deve riconoscere il carattere "*" conterrà "*". Ciò si applica indipendentemente dal carattere seguente, sia esso interpretabile come meta-carattere o meno. Nel caso in cui un carattere non alfanumerico debba identificare se stesso è opportuno farlo precedere dal "\". In particolare per identificare un backslash occorre scrivere "\\".

Se nel criterio di riconoscimento si specifica l’opzione PCRE_EXTENDED, lo spazio bianco (diversamente da quando si trova all’interno di una classe di caratteri), e i caratteri posti tra "#" e un "a capo" all’esterno di una classe di caratteri sono ignorati. Un backslash può essere usato come escape per inserire uno spazio bianco od il carattere "#" come parte del criterio di riconoscimento.

Un secondo utilizzo del backslash consiste nel codificare in modo visibile dei caratteri non visibili. Non ci sono restrizioni nella presenza di caratteri non-stampabili, a parte lo zero binario terminante la stringa dell’espressione regolare. Di seguito saranno elencate le sequenze di caratteri che è preferibile utilizzare per la loro semplicità al posto delle corrispondenti codifiche binarie.

\a

allarme, il carattere BEL (hex 07)

\cx

"control-x", dove x è un qualsiasi carattere

\e

escape (hex 1B)

\f

salto pagina (hex 0C)

\n

"a capo" (newline) (hex 0A)

\r

carriage return (hex 0D)

\t

tabulazione (hex 09)

\xhh

carattere il cui codice esadecimale è hh

\ddd

carattere il cui codice ottale è ddd, oppure riferimento all'indietro

Il preciso effetto di "`\cx`" è il seguente: se "`x`" è una lettera minuscola, viene convertita in lettera maiuscola. In pratica viene invertito il sesto bit (hex 40) del carattere. Quindi "`\cz`" diventa hex 1A, ma "`\c{`" diventa hex 3B, mentre "`\c;`" diventa hex 7B.

Dopo la sequenza "`\x`", saranno letti due numeri esadecimali (per le lettere non si distingue tra maiuscolo e minuscolo)

Dopo la sequenza "`\0`" saranno lette due cifre in ottale. In entrambi i casi se vi sono meno di due cifre, saranno usati i numeri presenti. Pertanto la sequenza "`\0\x\07`" indica 2 zeri binari seguiti dal carattere BEL. Occorre accertarsi di passare le cifre necessarie dopo lo zero iniziale se il carattere che segue può essere scambiato per una cifra in ottale.

Più complicata è la gestione del backslash seguito da una cifra diversa da 0. Al di fuori di una classe di caratteri, PCRE tratta le cifre che trova come numeri decimali. Se il numero è inferiore a 10, oppure vi sono state almeno altrettante parentesi sinistre, la sequenza viene considerata come un *riferimento all'indietro*. Più avanti, nella parte dei criteri parziali, sarà descritto come funzionano questi riferimenti.

All'interno di una classe di caratteri, oppure nel caso in cui il numero decimale è maggiore di 9 e non ci sono stati altrettanti criteri parziali, PCRE rilegge le prime 3 cifre seguenti il backslash in ottale e genera il carattere dagli 8 bit meno significativi del valore ottenuto. Ogni altra cifra seguente indica se stessa. Ad esempio:

\040

è un'altro modo per indicare uno spazio

\40
: ha il medesimo significato dell’esempio precedente che non vi sono 40 sotto-criteri

\7
: è sempre un riferimento all’indietro

\11
: può essere un riferimento all’indietro o un’altro modo per indicare il carattere di tabulazione

\011
: è ancora il carattere di tabulazione

\0113
: il carattere di tabulazione seguito da "3"

\113
: è il carattere con il codice ottale 113 (poiché non ci possono essere più di 99 riferimenti all’indietro)

\377
: è un byte con tutti i bit a 1

\81
: può essere un riferimento all’indietro o uno zero binario seguito da "8" e da "1"

Occorre rilevare che valori ottali maggiori di 100 non devono essere preceduti dallo zero, questo perché la libreria considera solo tre cifre.

Tutte le sequenze che definiscono il valore di un singolo byte possono essere utilizzate sia all’interno sia all’esterno delle classe di caratteri. Inoltre, all’interno delle classi di caratteri, la sequenza "`\b`" viene interpretata come carattere di backspace (hex 08), mentre all’esterno ha un’altro significato (come descritto più avanti).

Il terzo utilizzo possibile per il backslash consiste nello specificare il tipo di carattere:

\d
: qualsiasi cifra decimale

\D
: qualsiasi carattere che non sia una cifra decimale

\s
: qualsiasi carattere identificato come spazio bianco

\S
: qualsiasi carattere che non sia identificato come spazio bianco

\w
: qualsiasi carattere che sia una "parola" (word)

\W

qualsiasi carattere che non sia una "parola" (word)

Ciascuna coppia di sequenze di escape suddivide il set completo dei caratteri in due insiemi disgiunti. Un dato carattere deve essere identificato da un solo insieme di ciascuna coppia.

I caratteri definiti "parole" sono quelle lettere o cifre o il carattere underscore (_), cioè qualsiasi carattere che possa essere parte di una "`parola`" in Perl. In PCRE le definizioni di lettere e cifre vengono gestite tramite le tabelle dei caratteri, che possono variare in base a specifici parametri di localizzazione (vedere il paragrafo intitolato "Supporto alle localizzazioni"). Ad esempio, nella localizzazione fr (relativa alla Francia), qualche codice carattere maggiore di 128 è utilizzato per le lettere accentate, e queste sono identificate tramite la sequenza `\w`.

Queste sequenze di tipi di caratteri possono apparire sia all'interno sia all'esterno delle classi di caratteri. Ciascuna di esse identifica un carattere del tipo appropriato. Se durante la fase di identificazione di un testo, si giunge al termine della stringa in cui si esegue il riconoscimento e si hanno ancora di queste sequenze da incrociare, l'operazione di identificazione fallirà perché, ovviamente, non vi sono più caratteri in cui riconoscere le suddette sequenze.

Il quarto utilizzo per il backslash riguarda la costruzione di particolari asserzioni. L'asserzione è una condizione che deve essere soddisfatta ad un certo punto del riconoscimento, senza "consumare" caratteri dalla stringa oggetto del riconoscimento. Più avanti verranno descritte asserzioni più complicate, costruite tramite l'uso di sotto-criteri di riconoscimento, per ora saranno illustrate delle semplici asserzioni costruite con il backslash:

\b

limite di una parola

\B

non limite di una parola

\A

inizio dell'oggetto di ricerca (a prescindere dalla modalità multi-linea)

\Z

fine dell'oggetto di ricerca oppure newline alla fine (a prescindere dalla modalità multi-linea)

\z

fine dell'oggetto di ricerca (a prescindere dalla modalità multi-linea)

Queste asserzioni non possono apparire all'interno di una classe di caratteri (attenzione che la sequenza "`\b`" all'interno di una classe di caratteri indica il carattere backspace).

Viene definito limite di una parola la posizione nella stringa oggetto della ricerca, nella quale il carattere corrente ed il carattere precedente non soddisfano la sequenza `\w` o la sequenza `\W` (ad esempio uno soddisfa la sequenza `\w` e l'altro carattere soddisfa la sequenza `\W`), oppure quella posizione, all'inizio o alla fine della stringa, nella quale rispettivamente il primo o l'ultimo carattere soddisfa la sequenza \w.

Le asserzioni `\A`, `\Z` e `\z` differiscono dai tradizionali caratteri "^" e "$" (descritti di seguito) per il fatto di identificare sempre l'inizio o la fine della stringa oggetto di ricerca a prescindere da quale

opzione sia stata attivata. Infatti queste asserzioni non sono alterate da PCRE_NOTBOL oppure da PCRE_NOTEOL. La differenza tra `\Z` e `\z` consiste nel fatto che `\Z` identifica sia il carattere precedente il newline posto al termine della stringa sia la fine della stringa, mentre `\z` identifica solo la fine.

I caratteri "^" e "$"

In condizioni normali, il carattere "^", posto all'esterno di una classe di caratteri, indica un asserzione che è vera soltanto se il punto da cui si inizia a identificare il testo si trova all'inizio della stringa oggetto della ricerca. Al contrario, se il carattere "^" si trova all'interno di una classe di caratteri, assume altri significati (vedere i capitoli seguenti).

Non è necessario che il carattere "^" sia la prima lettera del criterio di riconoscimento nei casi in cui si utilizzino dei criteri di riconoscimento alternativi. Tuttavia è necessario che sia la prima lettera nei rami alternativi in cui compare. Se si inserisce il carattere "^" in tutte le alternative, caso che ricorre quando si vuole riconoscere un testo a partire dall'inizio della stringa, si dice che si è costruito un criterio di ricerca "ancorato". (Esistono anche altre costruzioni che portano all'ancoraggio di un criterio di ricerca).

Il carattere dollaro "$" è una asserzione che è `TRUE` soltanto se il punto a cui si è arrivati ad identificare il testo si trova alla fine della stringa oggetto della ricerca, o nella lettera immediatamente precedente il carattere di "a capo", che (per default) è l'ultimo carattere del testo. Il dollaro "$" non deve essere necessariamente l'ultima lettera di un criterio di riconoscimento se in questo si sono utilizzati dei criteri alternativi. Deve, comunque, essere l'ultima lettera nei criteri ove compare. Il carattere dollaro "$" non ha significati speciali all'interno di una classe di caratteri.

Il comportamento del simbolo "$" può essere variato in modo da identificare la reale fine della stringa oggetto di ricerca attivando il flag PCRE_DOLLAR_ENDONLY durante la compila o durante la fase di riconoscimento. Ciò non influisce sull'asserzione \Z.

Il comportamento dei simboli "^" e "$" può essere influenzato dall'opzione PCRE_MULTILINE. Quando viene attivata, questi caratteri identificano rispettivamente, oltre ai tradizionali inizio e fine testo, la lettera immediatamente successiva ed immediatamente precedente il carattere "\n" presente all'interno della stringa oggetto di ricerca. Per esempio il criterio /^abc$/ riconosce il testo "def\nabc" solo in modalità multi-linea. Di conseguenza i criteri di riconoscimento che sono "ancorati" in modalità singola linea, non lo sono in modalità multi-linea. L'opzione PCRE_DOLLAR_ENDONLY

viene ignorata se si attiva il flag PCRE_MULTILINE .

Occorre rilevare che le sequenze \A, \Z e \z possono essere
utilizzate per riconoscere l’inizio e la fine del testo in
entrambe le modalità, e, se tutti i criteri alternativi
iniziano con \A, si ha ancora un criterio "ancorato"
a prescindere che sia attivata o meno l’opzione PCRE_MULTILINE.

FULL STOP

Il carattere punto ".", all’esterno di una classe di caratteri,
identifica qualsiasi carattere, anche quelli non stampabili,
ma (per default) non il carattere "a capo". Invece se si
abilita l’opzione PCRE_DOTALL
si avrà anche il riconoscimento del carattere "a capo".
La gestione del simbolo "." è svincolata dalla gestione dei
simboli "^" ed "$", l’unica relazione che possono avere riguarda
il carattere "a capo". Il punto "." non ha significati speciali
all’interno delle classi di caratteri.

Parentesi quadrate

Un parentesi quadrata aperta "[" inizia una classe di caratteri;
una parentesi quadrata chiusa "]" termina la definizione
della classe. Di suo il carattere di parentesi quadrata chiusa
non ha significati speciali. Se occorre inserire la parentesi
chiusa all’interno di una classe di caratteri, questa deve
essere la prima lettera (ovviamente deve seguire il carattere "^",
se presente) oppure deve essere preceduta dal carattere di escape "\".

Una classe di caratteri identifica un singolo carattere nella
stringa oggetto di ricerca; il carattere deve comparire nel
set di caratteri definito dalla classe, a meno che il primo
carattere della classe non sia l’accento circonflesso "^",
in tal caso il carattere non deve essere nel set definito
dalla classe. Se è richiesto l’inserimento del carattere "^"
nel set definito dalla classe, questo non deve essere la
prima lettera dopo la parentesi di apertura, oppure deve
essere preceduto dal carattere di escape (\).

Ad esempio, la classe [aeiou] identifica ogni vocale minuscola,
mentre [^aeiou] identifica tutti i caratteri che non siano delle
vocali minuscole. Occorre notare che il simbolo "^" è un modo
pratico per indicare i caratteri che sono nella classe, citando
quelli che non lo sono. Questa non è una asserzione: consuma
un carattere della stringa oggetto di ricerca e fallisce se ci

si trova alla fine del testo.

In un riconoscimento senza distinzione tra minuscole e maiuscole, ogni lettera della classe identifica sia la versione maiuscola sia la minuscola. Così, ad esempio, la classe [aeiou] identifica sia "A" sia "a", e, in questo caso, [^aeiou] non identifica "A", mentre con la distinzione delle maiuscole [^aeiou] identifica la lettera "A".

Il carattere di "a capo" (newline) non viene trattato in modo speciale nelle classi di caratteri, indipendentemente dalle opzioni PCRE_DOTALL o PCRE_MULTILINE. La classe [^a] riconosce sempre il carattere "a capo".

Il segno meno (-) può essere usato per definire un intervallo all’interno della classe. Ad esempio, [d-m] identifica ogni lettera compresa tra d ed m inclusi. Se occorre inserire il segno meno (-) come carattere da riconoscere o lo si fa precedere dal carattere di escape (\), oppure lo si mette in una posizione tale che non possa essere identificato come definizione di un intervallo (ad esempio all’inizio o alla fine della definizione della classe).

Non è possibile usare il carattere "]" come limite di un intervallo. Un criterio definito come [W-]46], viene inteso come una classe di due caratteri (W e -) seguita dalla stringa 46], in tal modo sarebbero riconosciuti i testi "W46]" oppure "-46]". Quindi è necessario precedere la lettera "]" con il carattere di escape (\), in questo modo [W-\]46], viene interpretata correttamente come una singola classe contenente un range seguito da due caratteri separati. In alternativa, per delimitare l’intervallo si può utilizzare la notazione ottale di "]".

Gli intervalli utilizzano la sequenza di caratteri ASCII. Inoltre possono essere utilizzati per definire caratteri con specifica numerica (ad esempio [\000-\037]). Nei casi in cui si abiliti il riconoscimento senza distinzione tra lettere maiuscole e minuscole, gli intervalli comprendenti lettere identificano sia la lettera maiuscola che minuscola. Ad esempio, [W-c] è equivalente ad [][\^_‘wxyzabc] (con il riconoscimento a prescindere dalla lettera maiuscole e minuscole), e, se si utilizza la tabella dei caratteri locali francesi "fr", [\xc8-\xcb] identifica la lettera "e" accentata sia maiuscola sia minuscola.

Nelle classi di caratteri si possono utilizzare le sequenze \d, \D, \s, \S, \w e \W per aggiungere altri tipi di caratteri alla classe. Ad esempio, [\dABCDEF] riconosce qualsiasi cifra esadecimale. Il carattere "^" può essere utilizzato con i caratteri maiuscoli per indicare un set di caratteri più ristretto che l’identificazione del set di caratteri minuscoli. Ad esempio, la classe [^\W_] identifica qualsiasi lettera o cifra

ma non il trattino basso (_).

Tutti i caratteri non alfabetici, eccetto \, -, ^ (posto all'inizio) e] non sono speciali per la classi di caratteri, e non sono dannosi se preceduti dal caratteri di escape (\).

Barra verticale (|)

La barra verticale (|) viene utilizzata per separare criteri di riconoscimento alternativi. Ad esempio il seguente criterio

```
gilbert|sullivan
```

può riconoscere "gilbert" oppure "sullivan". Ci possono essere innumerevoli alternative, compresa un'alternativa vuota (per identificare una stringa vuota). Il processo di riconoscimento prova tutte le alternative possibili (da sinistra a destra) fino a quando non ne trova una che sia soddisfatta. Se le alternative sono nella definizione di un criterio parziale il riconoscimento ha successo quando avviene su tutto il criterio.

Settaggio delle opzioni interne

Le opzioni PCRE_CASELESS, PCRE_MULTILINE , PCRE_DOTALL e PCRE_EXTENDED possono essere cambiate "al volo" dall'interno del criterio di riconoscimento, utilizzando le sequenze di lettere definite per queste opzioni in Perl, racchiuse tra "(?" ed ")". Le lettere utilizzabili sono:

```
i  per PCRE_CASELESS
m  per PCRE_MULTILINE
s  per PCRE_DOTALL
x  per PCRE_EXTENDED
```

Ad esempio, (?im) abilita il riconoscimento senza distinzione tra lettere maiuscole e minuscole e la modalità multi-linea. E' anche possibile disabilitare queste opzioni facendo precedere la lettera dal segno meno (-), o combinare l'attivazione con la disattivazione come nel caso (?im-sx), in cui si attiva PCRE_CASELESS e PCRE_MULTILINE e si disattiva PCRE_DOTALL e PCRE_EXTENDED. Se la lettera compare sia prima che dopo il segno meno (-) l'opzione resta disabilitata.

L'ambito di validità delle opzioni dipende da dove sono i flag di abilitazione/disabilitazione nel criterio di ricerca. Nei casi in cui il settaggio di un'opzione avvenga all'esterno

di un qualsiasi criterio parziale (come definito in seguito) gli
effetti sono medesimi di quando l'attivazione/disattivazione
avviene all'inizio del criterio di riconoscimento. Gli esempi
seguenti si comportano tutti allo stesso modo:

```
(?i)abc
a(?i)bc
ab(?i)c
abc(?i)
```

In questi esempi si attiva PCRE_CASELESS
per il criterio "abc". In altre parole, questo settaggio compiuto
al primo livello si applica a tutto il criterio (a meno che non
vi siano altre variazioni nelle sotto-regole di riconoscimento).
Qualora vi siano più settaggi per lo stesso parametro, viene
utilizzato quello più a destra.

Se si inserisce la variazione di un'opzione in una sotto-regola
gli effetti sono differenti. Questa è una variazione rispetto al
comportamento di Perl 5.005. Una variazione apportata all'interno
di una sotto-regola, vale solo per la parte della sotto-regola
che segue la variazione, pertanto

```
(a(?i)b)c
```

identifica abc, aBc e nessuna altra stringa (si assume che
PCRE_CASELESS non sia usato).
Da ciò deriva che le opzioni possono avere comportamenti differenti
nelle varie parti del criterio di riconoscimento. Ogni variazione
apportata ad una ramo alternativo del criterio viene estesa alle
alternative successive della medesima sotto-regola. Ad esempio,

```
(a(?i)b|c)
```

identifica "ab", "aB", "c" e "C", anche se per identificare "C"
viene scartato il primo ramo prima di incontrare il settaggio
dell'opzione. Questo accade perché gli effetti dei settaggi
vengono inseriti nella fase di compila. Diversamente si avrebbero
dei comportamenti molto bizzarri.

Le opzioni specifiche di PCRE PCRE_UNGREEDY e PCRE_EXTRA
possono essere variate nel medesimo modo delle opzioni
Perl compatibili, utilizzando rispettivamente i caratteri
U e X. Il flag (?X) è particolare poiché deve comparire
prima di qualsiasi opzione esso attivi anche quando si
trova al primo livello. E' opportuno inserirlo all'inizio.

Sotto-regole

Le sotto-regole sono delimitate dalle parentesi tonde e possono essere annidate. La definizione di una parte di una regola di riconoscimento come sotto-regola può servire a:

1. Localizzare un set di alternative. Ad esempio nel seguente criterio di riconoscimento

```
cat(aract|erpillar|)
```

si riconosce una tra le parole "cat", "cataract" oppure "caterpillar". Se non fossero state usate le parentesi, il criterio avrebbe permesso il riconoscimento delle parole "cataract", "erpillar" oppure una stringa vuota.

2. Identificare e catturare parte del testo riconosciuto dalla sotto-regola (come illustrato in seguito). Quando il riconoscimento dell'intero criterio ha avuto successo, le porzioni della stringa oggetto della ricerca identificate dalle sotto-regole sono restituite alla funzione chiamante tramite l'argomento *vettore* della funzione **pcre_exec()**. Per determinare il numero ordinale di ciascuna sotto-regola si contano le parentesi aperte da sinistra verso destra partendo da 1.

Ad esempio, se si esegue un riconoscimento nella frase "the red king" con il seguente criterio

```
the ((red|white) (king|queen))
```

si ottengono i testi "red king", "red" e "king", rispettivamente identificati dalla sotto-regola 1, 2 e 3.

Il fatto che le parentesi tonde adempiano a due funzioni non sempre porta a situazioni comprensibili. Spesso capita di dovere raggruppare delle sotto-regole senza per questo essere richiesta la cattura del testo. A tale scopo l'uso della sequenza "?:" dopo la parentesi di apertura permette di indicare che questa sotto-regola non deve catturare il testo, e non deve essere conteggiata nel numero d'ordine delle sotto-regole di cattura. Ad esempio, applicando il criterio

```
the ((?:red|white) (king|queen))
```

alla frase "the white queen", si catturano i testi "white queen" e "queen", rispettivamente numerati 1 e 2. Il numero massimo di testi catturabili è 99, il numero massimo di sotto-regole, a prescindere che siano di cattura o meno, è 200.

Come utile abbreviazione, nei casi in cui l'attivazione di alcune opzioni debba essere fatta all'inizio di una sotto-regola

non di cattura, la lettera dell'opzione può comparire tra la "?" e ":". Pertanto i due criteri

```
(?i:saturday|sunday)
(?:(?i)saturday|sunday)
```

riconoscono esattamente lo stesso set di parole. Poiché i rami alternativi sono provati da sinistra verso destra, e le opzioni non sono azzerate fino a quando non si è conclusa la sotto-regola, una opzione attivata in un ramo alternativo si applica a tutte le alternative che seguono. Pertanto, nell'esempio di prima, sia la parola "SUNDAY" che la parola "Saturday" sono testi identificati correttamente.

Ripetizioni

Le ripetizioni sono il numero delle occorrenze che possono essere indicate dopo i seguenti elementi:

un singolo carattere, possibilmente preceduto dal carattere di escape (\)
il metacarattere .
una classe di caratteri
un riferimento all'indietro (vedere la sezione successiva)
una sotto-regola (a meno che non si tratti di una asserzione - vedere più avanti)

In generale una occorrenza specifica un numero minimo e massimo di riconoscimenti previsti tramite la specifica dei due limiti, posti tra parentesi graffe e separati da una virgola. Entrambi i numeri devono essere minori di 65536 ed il primo deve essere minore o uguale rispetto al secondo. Ad esempio:

```
z{2,4}
```

identifica "zz", "zzz" oppure "zzzz". La parentesi graffa chiusa di suo non rappresenta un carattere speciale. Se si omette il secondo numero, ma si lascia la virgola, non si indica un limite superiore; se sia la virgola sia il secondo numero sono omessi, l'occorrenza specifica l'esatto numero di riconoscimenti richiesti. Il criterio

```
[aeiou]{3,}
```

identifica almeno 3 o più vocali consecutive, mentre

```
\d{8}
```

identifica esattamente 8 cifre. Una parentesi graffa aperta che compaia in una posizione in cui non sia prevista una

occorrenza, o che comunque non sia riconducibile alla sintassi di una occorrenza, viene tratta come il carattere "{". Ad esempio {,6} non indica delle occorrenze, ma una stringa di 4 caratteri.

L'indicazione {0} è permessa. Indica di comportarsi come se l'elemento precedente all'occorrenza non fosse presente.

Per praticità (e compatibilità verso il passato) si usano 3 abbreviazioni per le occorrenze più comuni:

```
*    equivale a {0,}
+    equivale a {1,}
?    equivale a {0,1}
```

E' anche possibile creare un ciclo infinito facendo seguire ad una sotto-regola che non identifica alcun carattere una occorrenza priva di limite superiore. Ad esempio:

```
(a?)*
```

Le versioni precedenti di Perl e di PCRE segnalavano un errore durante la fase di compila per questi criteri. Tuttavia, poiché vi sono casi dove questi criteri possono essere utili, queste regole sono accettate, ma, se la ripetizione non riconosce alcun carattere, il ciclo viene interrotto.

Per default le occorrenze sono "bramose", infatti identificano più testi possibile (fino al numero massimo permesso), senza per questo fare fallire il riconoscimento della parte rimanente del criterio di ricerca. Il classico esempio dove questo comportamento può essere un problema, riguarda il riconoscimento dei commenti nei programmi C. Questi compaiono tra le sequenze /* ed */, ma all'interno, nel commento, si possono usare sia il carattere *, sia il carattere /. Il tentativo di individuare i commenti C con il seguente criterio

```
/\*.*\*/
```

se applicato al commento

```
/* primo commento */  nessun commento  /* secondo commento */
```

non ha successo, perché identifica l'intera stringa a causa della "bramosia" dell'elemento ".*".

Tuttavia, se un'occorrenza è seguita da un punto interrogativo, questa cessa di essere "bramosa", e identifica il numero minimo di testi permessi. Pertanto il criterio

```
/\*.*?\*/
```

si comporta correttamente con i commenti C. Il significato delle varie occorrenze non è stato cambiato, si è soltanto modificato il numero delle occorrenze da riconoscere. Attenzione a non confondere questo uso del punto interrogativo con il suo proprio significato. Dati i due utilizzi del ?, può anche capitare di averli entrambi come in

```
\d??\d
```

che, preferibilmente, identifica una cifra, ma può riconoscerne due se questa è l'unica via per soddisfare il riconoscimento dell'intero criterio.

Se si attiva l'opzione PCRE_UNGREEDY (un'opzione disponibile anche in Perl), per default le occorrenze non sono "bramose", ma ogni singola occorrenza può essere resa "bramosa" se seguita dal punto interrogativo. In altre parole si è invertito il normale comportamento.

Quando si indica un numero minimo di occorrenze maggiore di 1, per una sotto-regola posta tra parentesi, oppure si indica un numero massimo di occorrenze, la fase di compila richiede più spazio in proporzione ai valori minimo e massimo.

Se un criterio inizia con .* oppure .{0,} e si è attivata l'opzione PCRE_DOTALL (equivalente al /s di Perl), permettendo al . di identificare il carattere di "a capo", si dice che il criterio è implicitamente ancorato, perché qualsiasi elemento che segue sarà confrontato con ciascun carattere della stringa oggetto della ricerca, e pertanto non vi è nessuna posizione, a parte la prima, da cui ripartire per ritentare il riconoscimento dell'intero criterio. PCRE tratta tale criterio come se fosse preceduto dalla sequenza \A. Nei casi in cui è noto a priori che la stringa oggetto di ricerca non contiene caratteri di "a capo", vale la pena di attivare l'opzione PCRE_DOTALL se il criterio di ricerca inizia con ".*", per ottenere questa ottimizzazione, oppure, in alternativa, usare "^" per indicare in modo esplicito un ancoraggio.

Quando si ripete una sotto-regola di cattura, il testo estrapolato è la parte identificata dall'iterazione finale. Ad esempio se il criterio

```
(tweedle[dume]{3}\s*)+
```

viene applicato al testo "tweedledum tweedledee", il testo estrapolato sarà "tweedledee". Tuttavia se vi sono delle sotto-regole annidate, il testo catturato può essere determinato dalle iterazioni precedenti. Ad esempio nel criterio

```
/(a|(b))+/
```

applicato a "aba", la seconda stringa catturata è "b".

Riferimenti all'indietro

Esternamente ad una classe di caratteri, un backslah (\) seguito da una o più cifre maggiori di 0 è un riferimento all'indietro verso testi catturati precedentemente (ad esempio alla sinistra rispetto a dove ci si trova), conteggiati tramite il numero delle parentesi chiuse precedenti.

Tuttavia, se il numero che segue il backslash (\) è un valore inferiore a 10, si assume che si tratti sempre di un riferimento all'indietro, e pertanto viene generato un errore se nel criterio non vi sono almeno altrettante parentesi chiuse di sotto-regole di cattura. In altre parole per i numeri inferiori a 10 non è necessario che il numero parentesi precedenti (a sinistra) alla sotto-regola sia pari o maggiore al numero indicato. Vedere la sezione relativa al backslash per informazioni su come gestire le cifre seguenti il backslash.

Un riferimento all'indietro identifica ciò che è già stato catturato dalla sotto-regola, e non ciò che la sotto-regola stessa possa riconoscere. Ad esempio il criterio

```
(sens|respons)e and \1ibility
```

può riconoscere "sense and sensibility" oppure "response and responsibility", ma non il testo "sense and responsibility".
Se al momento del riferimento all'indietro, è attiva la distinzione tra lettere maiuscole e minuscole, il formato della lettera diventa rilevante. Ad esempio,

```
((?i)rah)\s+\1
```

può identificare "rah rah" e "RAH RAH", ma non "RAH rah", anche se la sotto-regola originale eseguiva dei riconoscimenti a prescindere dalle lettere maiuscole e minuscole.

Nella medesima sotto-regola si possono avere più di un riferimento all'indietro. Se una sotto-regola non viene utilizzata in un particolare riconoscimento, un riferimento a questa ha sempre esito negativo. Ad esempio il criterio

```
(a|(bc))\2
```

non avrà mai successo se l'identificazione della stringa
inizia con "a" e non con "bc". Dato che sono possibili fino a 99
riferimenti all'indietro, tutte le cifre che seguono un backslash
(\) sono considerate come parte di un potenziale numero di
riferimento. Se il criterio continua con altri caratteri
numerici, occorre stabilire un carattere per delimitare il
numero del riferimento. Se si attiva l'opzione PCRE_EXTENDED
questo carattere può essere lo spazio. Altrimenti si può usare
un commento vuoto.

Un riferimento all'indietro non ha successo se viene inserito
in una sotto regola prima che sia applicata. Con questo si
intende, ad esempio, che (a\1) non avrà mai successo, ma, nel
caso di una sottoregola ripetuta, si avrà un riscontro positivo.
Ad esempio

```
(a|b\1)+
```

identificherà qualsiasi numero di "a", ma anche "aba" oppure
"ababaa" eccetera. In pratica a ciascuna iterazione della
sotto-regola il riferimento andrà a sostituire la stringa riconosciuta
tramite la sotto-regola dell'iterazione precedente. Affinchè
il tutto funzioni, è necessario che la prima iterazione sia
identificata senza l'ausilio del riferimento "all'indietro".
Ciò può essere ottenuto o usando casi alternativi, come nell'esempio
precedente, o usando le occorrenze indicando come numero minimo
di occorrenze il valore zero.

Asserzioni

L'asserzione è un test basato su un carattere, che può precedere
o seguire l'attuale punto di riconoscimento, e non consuma
alcun carattere. Le semplici asserzioni quali \b, \B, \A, \Z,
\z, ^ e $ sono state descritte precedentemente. Asserzioni più
complesse possono essere strutturate come delle sotto-regole.
Se ne hanno di due tipologie: quelle che "guardano avanti"
alla posizione attuale nella stringa oggetto del riconoscimento,
e quelle che "guardano dietro" la posizione attuale.

Una asserzione definita come sotto-regola esegue il riconoscimento
nel modo usuale, ma tale riconoscimento non sposta la posizione
attuale nella stringa. Le asserzioni che "guardano avanti" cominciano
per "(?=", se sono positive, per "(?!", se sono asserzioni negative.
Ad esempio

```
\w+(?=;)
```

riconosce una parola seguita da ";", ma non include il
punto e virgola nel riconoscimento, mentre

```
foo(?!bar)
```

identifica qualsiasi occorrenza di "foo" che non sia seguita da "bar". Attenzione che il criterio, apparentemente simile,

```
(?!foo)bar
```

non riconosce alcuna occorrenza di "bar" se questa è preceduta da qualsiasi testo che non sia "foo"; infatti l'espressione riconosce qualsiasi occorrenza di "bar", poiché l'asserzione (?!foo) è sempre `TRUE` quando i tre caratteri successivi sono "bar". Pertanto è necessario una asserzione che "guarda" all'indietro per ottenere effetto desiderato.

Le asserzioni che "guardano" indietro positive iniziano con "(?<=", e con "(?<!" le negative. Ad esempio:

```
(?<!foo)bar
```

riconosce una occorrenza di "bar" che non sia preceduta da "foo". Le asserzioni che "guardano" indietro hanno una limitazione: tutte le stringhe che riconoscono devono avere lunghezza fissa. Mentre, se si hanno casi alternativi, la limitazione della lunghezza fissa non sussiste. Quindi

```
(?<=bullock|donkey)
```

è una asserzione permessa, ma

```
(?<!dogs?|cats?)
```

genera un errore durante la fase di compila. Rami alternativi con lunghezze di stringa differenti sono permessi solo al primo livello dell'asserzione. Questa è da considerarsi una estensione rispetto a Perl 5.005, che richiede a tutte le alternative possibili la medesima lunghezza di stringa. Quindi una asserzione tipo

```
(?<=ab(c|de))
```

non è permessa, poiché il suo singolo ramo di primo livello può identificare testi di due lunghezze differenti, ma è accettabile se riscritta usando due alternative di primo livello:

```
(?<=abc|abde)
```

L'implementazione di questo tipo di asserzioni consiste, per ciascuna alternativa, di uno spostamento all'indietro temporaneo per la lunghezza fissa necessaria, e ad un tentativo di riconoscimento del testo. Se non ci sono sufficienti caratteri precedenti alla posizione attuale, l'asserzione è destinata a

fallire. L’uso accoppiato delle asserzioni che "guardano" indietro con sotto-regole a riconoscimento singolo può essere utile per identificare la fine dei testi; un esempio è illustrato al termine della sezione sulle sotto-regole a riconoscimento singolo.

Diverse asserzioni (di qualsiasi tipologia) possono essere utilizzate in modo consecutivo. Ad esempio:

```
(?<=\d{3})(?<!999)foo
```

riconosce "foo" preceduto da tre cifre che non siano "999". Occorre rilevare che ciascuna asserzione viene applicata singolarmente sul medesimo punto nella stringa oggetto di riconoscimento. Nell’esempio precedente per prima cosa si verifica che i tre caratteri precedenti siano cifre, quindi che non siamo "999". Questo esempio non identifica il testo se "foo" è preceduto da sei caratteri di cui i primi tre siano cifre e i secondi tre non siano "999". In pratica il testo "123abcfoo" non viene riconosciuto. Un criterio per riconoscere tale stringa può essere:

```
(?<=\d{3}...)(?<!999)foo
```

In questo caso la prima asserzione controlla i primi sei caratteri verificando che i primi tre siano cifre, mentre la seconda asserzione verifica che i secondi tre caratteri non siano "999".

Le asserzioni possono essere annidate in qualsiasi combinazione. Il seguente esempio

```
(?<=(?<!foo)bar)baz
```

identifica il testo "baz" se è preceduto da "bar" il quale non deve essere preceduto da "foo", mentre

```
(?<=\d{3}(?!999)...)foo
```

è un’altro criterio che riconosce "foo" preceduto da tre cifre e da tre caratteri che non siano "999".

Le asserzioni definite come sotto-regole non catturano parte del testo e non possono essere ripetute (non avrebbe senso ripetere il medesimo riconoscimento sul medesimo testo). Se una di queste asserzioni contiene una sotto-regola di cattura questa viene conteggiata ai fini della numerazione delle regole di cattura. Tuttavia il testo viene effettivamente catturato solo nelle asserzioni positive, dato che non avrebbe senso farlo in quelle negative.

Le asserzioni possono essere composte fino ad un massimo

di 200 sotto-regole.
subpatterns.

Sotto-regole a riconoscimento singolo (Once-only subpatterns)

Con l'indicazione del numero minimo e massimo di ripetizioni, il fallimento del riconoscimento della parte successiva del testo causa una ripetizione dell'identificazione con un numero di occorrenze diverse per verificare se in questa situazione anche il resto del testo viene identificato. In alcune situazioni può essere utile bloccare questo meccanismo, per la cambiata natura del criterio, oppure per fare fallire la ricerca prima di quando potrebbe accadere, oppure quando l'autore sa che non ci sono punti da cui ripartire.

Consideriamo, ad esempio, il criterio \d+foo applicato alla seguente linea

```
123456bar
```

Dopo avere identificato le 6 cifre ed avere mancato nel riconoscimento di "foo", il comportamento normale consiste nel tentare di procedere identificando 5 cifre per l'elemento \d+, quindi tentare con 4 e così via, prima di fallire definitivamente.
Le sotto-regole a riconoscimento singolo permettono di specificare che, una volta riconosciuta una porzione della regola, questa non debba essere più considerata, in maniera tale da abortire immediatamente il riconoscimento se non si ha successo nell'identificare la parte restante del criterio.
L'espressione per definire questo tipo di sotto-regola richiede un'altro tipologia di utilizzo speciale delle parentesi. Infatti iniziano con (?> come nell'esempio seguente:

```
(?>\d+)bar
```

Questa tipologia di parentesi "inchioda" la parte di criterio una volta che avviene il riconoscimento, e, quindi, un fallimento successivo impedisce la ri-elaborazione di questo segmento. Tuttavia, occorre notare che gli elementi precedenti a questo si comportano in modo normale, e pertanto la ri-elaborazione, pur non toccando questo elemento, passa ai precedenti.

Una descrizione alternativa di questa tipologia di sotto-regola potrebbe essere che questo criterio identifica un testo come avrebbe fatto un singolo criterio se fosse stato ancorato alla posizione corrente.

Le sotto-regole a riconoscimento singolo non compiono la cattura del testo identificato. I casi semplici illustrati in precedenza

possono essere considerati come una estremizzazione della ripetizione che porta ad inglobare tutto ciò che può. Pertanto, da una parte \d+ e \d+? sono sequenze che si adattano a riconoscere il numero corretto di cifre affinchè la ricerca abbia successo, dall'altra la sequenza (?>\d+) riconosce soltanto una sequenza di cifre.

Ovviamente queste costruzioni possono contenere diverse sotto-regole sia complesse, sia annidate.

Le sotto-regole a riconoscimento singolo possono essere usate congiuntamente alle asserzioni che guardano indietro, per definire una regola efficiente per riconoscere la fine della stringa. Ad esempio si consideri questo semplice criterio:

```
abcd$
```

quando viene applicato ad un lungo testo può non avere successo. Questo perché il riconoscimento procede da sinistra verso destra, quindi PCRE prima cerca la "a", e poi cerca di riconoscere la parte restante del criterio. Se si modifica il criterio nel seguente modo

```
^.*abcd$
```

allora la sequenza iniziale .* in prima battuta identificherà tutto il testo, ma quando fallisce (poiché non vi sono più "a"), la ricerca tornerà indietro di uno (quindi tutto il testo tranne l'ultimo carattere), quindi, se continua non esserci la "a", si torna indietro di due, e così via. Continuando a tornare indietro alla ricerca della "a" si percorre tutto il testo da destra a sinistra, senza ottenere nulla di valido. Tuttavia se si riscrive il criterio come:

```
^(?>.*)(?<=abcd)
```

non si attiva più lo scorrimento verso sinistra per .* , ma viene costretto a riconoscere tutto il testo (esito del primo tentativo). La asserzione successiva, esegue una verifica sulle ultime 4 lettere.Se fallisce, ciò avviene immediatamente, provocando un impatto sensibile nei tempi di elaborazione con testi molto lunghi.

Quando un criterio di riconoscimento contiene un elemento ripetuto senza limite all'interno di una sotto-regola che anch'essa possa ripetuta illimitatamente, l'uso delle sotto-regole a riconoscimento singolo è l'unico mezzo per evitare che certi mancati riconoscimenti richiedano molto tempo per essere rilevati. Ad esempio il criterio

```
(\D+|<\d+>)*[!?]
```

riconosce un numero indefinito di frammenti di testo contenenti a loro volta numeri o caratteri non numerici racchiusi tra <>, seguiti dai caratteri ! o ?. Quando il riconoscimento ha successo l'esecuzione è rapida, ma quando viene applicato al testo

```
aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa
```

richiede molto tempo prima di evidenziare l'errore. Questo accade perché la stringa può essere suddivisa tra i due elementi ripetuti in un elevato numero di modi che debbono essere verificati. Nell'esempio si usa [!?] piuttosto che un singolo carattere alla fine, questo perché sia Perl sia PCRE hanno una ottimizzazione che permette un veloce riconoscimento dell'errore se si usa un solo carattere. Infatti memorizzano l'ultimo carattere singolo richiesto per un riconoscimento, e, se non lo trovano nel testo, falliscono subito. Se il criterio viene modificato in

```
((?>\D+)|<\d+>)*[!?]
```

le sequenze di caratteri non numerici non possono essere interrotte e pertanto il mancato riconoscimento viene rilevato più velocemente.

Sotto-regole condizionali (Conditional subpatterns)

E' possibile forzare il processo di riconoscimento a obbedire alle sotto-regole in modo condizionato, oppure a scegliere tra due alternative in base al risultato di una asserzione, o piuttosto se la sotto-regola di cattura precedente ha riconosciuto il proprio testo. I due metodi possibili per esprimere una sotto-regola condizionale sono:

```
(?(condizione)yes-pattern)
(?(condizione)yes-pattern|no-pattern)
```

Se la condizione è soddisfatta, si usa la regola indicata in yes-pattern, altrimenti si applica il no-pattern (qualora sia specificato). Se nella sotto-regola vi sono più di due alternative, PCRE segnala un errore in fase di compila.

Vi sono due tipi di condizioni. Se il testo tra le parentesi consiste in una sequenza di cifre, allora la condizione è soddisfatta se la sotto-regola di cattura di quel numero è stata precedentemente riconosciuta. Si consideri il seguente criterio contenente degli spazi non significativi per renderlo più leggibile (si assuma che sia attiva l'opzione PCRE_EXTENDED) e lo si divida in tre parti per praticità di discussione:

```
( \( )?    [^()]+    (?(1) \) )
```

La prima parte riconosce una parentesi aperta opzionale, e, se è presente, indica quello come primo segmento di stringa catturato. La seconda parte riconosce uno o più caratteri che non siano parentesi. Infine la terza parte è una sotto-regola condizionale che verifica se la prima parentesi è stata identifica o meno. Se accade, come nel caso in cui il testo inizi con una parentesi aperta, la condizione è `TRUE`, e quindi viene eseguito il ramo yes-pattern, richiedendo, conseguentemente, la parentesi chiusa. Diversamente, poiché non è specificato ramo no-pattern, non si esegue nessun altro riconoscimento. In altre parole questo criterio identifica un testo che possa essere racchiuso tra parentesi opzionali.

Se la condizione non è una sequenza di cifre, deve essere una asserzione. Questa può essere sia una asserzione che guarda avanti, sia una asserzione cha guarda indietro, sia positiva sia negativa. Si consideri il seguente criterio, ancora una volta contenente spazi non significativi, e con due alternative nella seconda linea:

```
(?(?=[^a-z]*[a-z])
\d{2}-[a-z]{3}-\d{2}  |  \d{2}-\d{2}-\d{2} )
```

La condizione è una asserzione che guarda avanti positiva, la quale identifica una sequenza opzionale di caratteri non alfabetici seguiti da una lettera. In altre parole, si testa la presenza di almeno una lettera nel testo. Se la lettera viene trovata si applica il primo criterio per il riconoscimento del testo, altrimenti viene applicato il secondo criterio. Questo criterio riconosce stringhe in due formati dd-aaa-dd oppure dd-dd-dd, dove aaa sono lettere e dd sono numeri.

Commenti

La sequenza (?# indica l'inizio di un commento che si conclude con la successiva parentesi chiusa. Parentesi annidate non sono permesse. I caratteri che fanno parte di un commento non giocano alcun ruolo nella fase di riconoscimento.

Se viene attivata l'opzione PCRE_EXTENDED, un carattere # privo della sequenza di escape (\) all'esterno di una classe di caratteri apre un commento che continua fino al carattere di "a capo".

Criteri ricorsivi

Si consideri il caso in cui si debba riconoscere una stringa tra parentesi, si supponga inoltre di dovere ammettere un numero arbitrario di parentesi annidate. Senza l'uso della ricorsione, il miglior metodo consiste nel preparare un criterio che identifichi un numero finito di parentesi annidate. Però, in questo modo non si gestisce un numero arbitrario si parentesi annidate. Nella versione 5.6 di Perl è stata introdotta, a livello sperimentale, la possibilità per i criteri di riconoscimento di essere ricorsivi. Allo scopo è stata definita la sequenza speciale (?R). Il criterio illustrato di seguito risolve il problema delle parentesi (si assume che l'opzione PCRE_EXTENDED sia attivata in modo da ignorare gli spazi):

```
\( ( (?>[^()]+) | (?R) )* \)
```

In principio si identifica la parentesi di apertura. Quindi si cerca un numero indefinito di stringhe che possano essere composte da caratteri che non siano parentesi, oppure che siano riconosciute ricorsivamente dal criterio (ad esempio una stringa correttamente tra parentesi). Infine si cerca la parentesi di chiusura.

In questo particolare esempio si hanno arbitrarie ripetizioni annidate, e quindi l'uso delle sotto-regole a riconoscimento singolo per il riconoscimento della stringa priva di parentesi è particolarmente utile se il criterio viene applicato ad un testo non riconoscibile. Ad esempio, applicando il criterio a

```
(aaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaaa()
```

si ottiene un esito di mancato riconoscimento rapidamente. Se, al contrario, non si fosse usato il criterio a riconoscimento singolo, sarebbe occorso molto tempo per avere un esito dato che vi sono moltissimi modi in cui i caratteri di ripetizione + e * possono essere applicati al testo, richiedendo, pertanto, la verifica di tutte le combinazioni prima di fornire l'esito negativo.

Il valori restituiti da una sotto-regola di cattura sono quelli ottenuti dal livello di ricorsione più esterno. Se il criterio illustrato in precedenza venisse applicato al testo

```
(ab(cd)ef)
```

Il valore ottenuto sarebbe "ef", che rappresenta l'ultimo valore catturato al livello più alto. Se si aggiungono altre parentesi al criterio, ottenendo

```
\( ( ( (?>[^()]+) | (?R) )* ) \)
   ^                    ^
```

^ ^ allora il testo ottenuto sarebbe
"ab(cd)ef", cioè il valore delle parentesi di livello più alto.
Se nel criterio vi sono più di 15 sotto-regole di cattura,
PCRE ha necessità di ottenere maggiore memoria per archiviare
le informazioni durante la ricorsione. Questa memoria è ottenuta
utilizzando pcre_malloc, al termine verrà liberata con pcre_free.
Se non può essere ottenuta ulteriore memoria, si ha il salvataggio
delle informazioni dei primi 15 testi catturati, dato che non vi è modo
di restituire un messaggio di memoria insufficiente dalla
ricorsione.

Performances

Certi elementi utilizzati per i criteri di riconoscimento sono più
efficienti di altri. E' più efficiente usare le classi di caratteri come
[aeiou] piuttosto che un gruppo di alternative come (a|e|i|o|u).
In generale la costruzione più semplice per ottenere un dato
scopo è la più efficiente. Nel libro di Jeffrey Friedl sono
illustrate varie tecniche sull'ottimizzazione delle espressioni
regolari.

Quando un criterio comincia con .* ed è attiva l'opzione PCRE_DOTALL, il
criterio è implicitamente ancorato da PCRE, dato che può rico-
noscere solo l'inizio della stringa. Tuttavia, se non è attivo
PCRE_DOTALL, PCRE non può
fare questa ottimizzazione, perché il meta-carattere . non ri-
conosce più il carattere di "a capo", e quindi se la stringa
contiene dei caratteri di "a capo", il riconoscimento può par-
tire dal carattere immediatamente successivo ad uno di questi
e non dall'inizio. Ad esempio il criterio

(.*) second

può eseguire un riconoscimento nel testo "first\nand second"
(dove \n indica il carattere "a capo") ottenendo "and" come
prima stringa catturata. perché ciò accada è necessario che
PCRE possa iniziare il riconoscimento dopo ogni "a capo".

Se si deve usare un simile criterio in stringhe che non conten-
gono caratteri di "a capo", le performance migliori si possono
ottenere abilitando PCRE_DOTALL,
oppure iniziando il criterio con ^.* in modo da richiedere un
ancoraggio esplicito. Così si risparmia a PCRE di scandirsi il
testo alla ricerca di un "a capo" da cui ripartire per la ri-
cerca.

Occorre prestare attenzione ai criteri che contengono ripeti-
zioni indefinite annidate. Possono richiedere molto tempo se
applicati a stringhe non riconoscibili. Si consideri il fram-

mento

(a+)*

Questo può riconoscere "aaaa" in 33 modi differenti, e questo numero può crescere molto rapidamente se la stringa da riconoscere è più lunga. (Il carattere di ripetizione * può eseguire riconoscimenti per 0,1,2,3 o 4 ripetizioni, e per ciascuna di esse, tranne lo 0, il carattere di ripetizione + può eseguire riconoscimenti per diversi numeri di volte). Quindi se la parte successiva del criterio è tale da non permettere il completamento del riconoscimento, PCRE, per principio, ugualmente tenta tutte le possibili variazioni, richiedendo diverso tempo.

Una ottimizzazione riesce a catturare qualche caso semplice come in

(a+)*b

dove si indica che seguirà un carattere alfanumerico. PCRE, prima di imbarcarsi nella procedura standard di riconoscimento, verifica se nel testo vi è la lettera "b", e se non c'è restituisce immediatamente un esito negativo. Tuttavia se non viene indicata una lettera seguente questa ottimizzazione non può essere utilizzata. Se ne può notare la differenza analizzando il comportamento del criterio

(a+)*\d

rispetto al precedente. Il primo, utilizzato con una stringa composta da "a", fallisce immediatamente, l'ultimo richiede un tempo apprezzabile, soprattutto se applicato a stringhe con più di 20 caratteri.

preg_grep (PHP 4 >= 4.0.0)

Restituisce una matrice degli elementi riconosciuti tramite le espressioni regolari

array **preg_grep** (string espressione_regolare, array testo) \linebreak

La funzione **preg_grep()** restituisce una matrice composta dagli elementi dell'array *testo* che soddisfano i criteri impostati nel parametro *espressione_regolare*.

A partire dalla versione 4.0.4 di PHP, la matrice risultante dalla funzione **preg_grep()**, viene indicizzata utilizzando le chiavi dalla matrice di input. Se non si desidera un comportamento simile, applicare la funzione array_values() alla matrice ottenuta da questa funzione per ricalcolare gli indici.

Esempio 1. Esempio di preg_grep()

```
// esempio di restituzione di tutti gli elementi della matrice
// contenenti numeri in virgola mobile
$fl_array = preg_grep ("/^(\d+)?\.\d+$/", $array);
```

preg_match (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Riconoscimento con espressioni regolari

int **preg_match** (string criterio, string testo [, array testi_riconosciuti]) \linebreak

Esegue un riconoscimento nel parametro *testo* utilizzando l'espressione regolare indicata in *criterio*.

Se viene fornito il terzo parametro, *testi_riconosciuti*, questo verrà valorizzato con i risultati della ricerca. In dettaglio $testi_riconosciuti[0] conterrà il testo che si incrocia con l'intero criterio di ricerca, $testi_riconosciuti[1] conterrà il testo che soddisfa il primo criterio posto tra parentesi, $testi_riconosciuti[2] il secondo e così via.

La funzione **preg_match()** restituisce il numero di volte in cui è avvenuto il riconoscimento del *criterio*. Questo può essere 0 (nessun riconoscimento) oppure 1 se **preg_match()** si ferma dopo il primo riconoscimento. In condizioni normali, preg_match_all() continua il riconoscimento fino alla fine del parametro *testo*. **preg_match()** restituirà FALSE se si verifica un errore.

Esempio 1. Ricerca del testo "php"

```
// La lettera "i" dopo i delimitatori indica una ricerca case-insensitive
if (preg_match ("/php/i", "PHP è il linguaggio scelto.")) {
    print "Il riconoscimento è avvenuto.";
} else {
    print "Testo non riconosciuto.";
}
```

Esempio 2. Cerca la parola "web"

```
// La lettera \b nel criterio indica i limiti della parola. Così verrà riconosci-
uta solo
// la parola "web" e non parte di una parola più lunga come "webbing" oppure "cobweb"
if (preg_match ("/\bweb\b/i", "PHP è un linguaggio di programmazione per il web scelto da
    print "Il riconoscimento è avvenuto.";
} else {
    print "Testo non riconosciuto.";
}
```

```
if (preg_match ("/\bweb\b/i", "PHP è un linguaggio di programmazione instal-
lato su molti website")) {
    print "Il riconoscimento è avvenuto.";
} else {
    print "Testo non riconosciuto.";
}
```

Esempio 3. Estrapolazione del dominio da un URL

```
// come ottenere il nome dell'host da un URL
preg_match("/^(http:\/\/)?([^\/]+)/i",
"http://www.php.net/index.html", $matches);
$NomeHost = $matches[2];
// come ottenere gli ultimi due segmenti del nome dell'host
preg_match("/[^\.\/]+\.[^\.\/]+$/",$NomeHost,$matches);
echo "Nome del dominio: ".$matches[0]."\n";
```

L'esempio visualizzerà:

```
Nome del dominio: php.net
```

Vedere anche preg_match_all(), preg_replace() e preg_split().

preg_match_all (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Esegue un riconoscimento globale con le espressioni regolari

int **preg_match_all** (string espressione_regolare, string testo, array TestiRiconosciuti [, int ordine]) \linebreak

La funzione ricerca tutte le espressioni regolari passate nel parametro *espressione_regolare* all'interno della stringa *testo*. I testi riconosciuti sono posti all'interno della matrice *TestiRiconosciuti*, nell'ordine specificato da *ordine*.

Dopo avere riconosciuto il primo segmento di testo, le ricerche seguenti saranno effettuate a partire dall'ultima ricerca specificata.

ordine può assumere uno dei due valori seguenti:

PREG_PATTERN_ORDER

I testi riconosciuti saranno organizzati in modo tale da avere in $TestiRiconosciuti[0] la matrice di tutti i testi riconosciuti, in $TestiRiconosciuti[1] la matrice di tutti i testi che

soddisfano il primo criterio di riconoscimento posto tra parentesi tonde, in $TestiRiconosciuti[2] si avranno i testi che soddisfano il secondo criterio e cosi via.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_PATTERN_ORDER);
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n";
```

Questo esempio produrrà:

```
<b>example: </b>, <div align=left>this is a test</div>
example: , this is a test
```

Nell’esempio, $out[0] contiene la matrice di tutte le stringhe che soddisfano i criteri impostati, $out[1] contiene la matrice dei testi delimitati dai tag.

PREG_SET_ORDER

Usando questo parametro come ordine dei riconoscimenti, si avrà in $TestiRiconosciuti[0] una matrice con il primo set di testi riconosciuti, in $TestiRiconosciuti[1], la matrice con il secondo set di testi riconosciuti e così via.

```
preg_match_all ("|<[^>]+>(.*)</[^>]+>|U",
    "<b>example: </b><div align=left>this is a test</div>",
    $out, PREG_SET_ORDER);
print $out[0][0].", ".$out[0][1]."\n";
print $out[1][0].", ".$out[1][1]."\n";
```

Questo esempio visualizzerà:

```
<b>example: </b>, example:
<div align=left>this is a test</div>, this is a test
```

In questo esempio $out[0] è la matrice del primo set di testi riconosciuti. Nel dettaglio $out[0][0] conterrà il testo che incrocia l’intero criterio impostato, $out[0][1] conterrà il testo riconosciuto tramite il prima regola di riconoscimento presente nel criterio globale. Analogo discorso si può fare per $out[1]. In questo caso il ragionamento verrà svolto su un secondo segmento della stringa che soddisfi le condizioni poste dal criterio di riconoscimento.

Qualora non si specifichi il parametro `ordine`, si assume per default il valore PREG_PATTERN_ORDER.

La funzione restituisce il numero dei riconoscimenti completi svolti (che possono essere zero), oppure FALSE se si verificano degli errori.

Esempio 1. Esempio di come ottenere tutti i numeri di telefono da un testo.

```
preg_match_all ("/\(?  (\d{3})?  \)?  (?(1)  [\-\s] ) \d{3}-\d{4}/x",
                "Call 555-1212 or 1-800-555-1212", $numeri);
```

Esempio 2. Ricerca di tag HTML

```
// Il parametro \\2 è un esempio di riferimento all'indietro. Questo dato in-
forma la
// libreria pcre che deve considerare il secondo set di parentesi tonde (in questo
// caso il testo "([\w]+)"). Il backslash (\) aggiuntivo è reso obbligatorio dall'uso
// dei doppi apici.
$html = "<b>bold text</b><a href=howdy.html>click me</a>;

preg_match_all ("/(<([\w]+)[^>]*>)(.*)(<\/\\2>)/", $html, $matches);

for ($i=0; $i< count($matches[0]); $i++) {
   echo "intero criterio: ".$matches[0][$i]."\n";
   echo "parte 1: ".$matches[1][$i]."\n";
   echo "parte 2: ".$matches[3][$i]."\n";
   echo "parte 3: ".$matches[4][$i]."\n\n";
}
```

Questo esempio visualizzerà:

```
intero criterio: <b>bold text</b>
parte 1: <b>
parte 2: bold text
parte 3: </b>

intero criterio: <a href=howdy.html>click me</a>
parte 1: <a href=howdy.html>
parte 2: click me
parte 3: </a>
```

Vedere anche preg_match(), preg_replace() e preg_split().

preg_quote (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Inserisce il carattere di escape nei caratteri delle espressioni regolari

string **preg_quote** (string stringa [, string delimitatori]) \linebreak

La funzione **preg_quote()** inserisce il carattere di escape (\) davanti ad ogni carattere presente in `stringa` che sia parte della sintassi di una espressione regolare. Questa funzione è utile nei casi in cui si generino, durante l'esecuzione, delle stringhe da usare come criteri di riconoscimento, e queste possano contenere dei caratteri speciali per le espressioni regolari.

Se viene specificato un carattere come parametro *delimitatore*, anche a questo sarà anteposto il carattere di escape (\). Ciò è particolarmente utile per porre il carattere di escape nei delimitatori richiesti dalle funzioni PCRE. Il carattere di delimitazione più comunemente utilizzato è /.

I caratteri speciali per le espressioni regolari sono:

```
. \\ + * ? [ ^ ] $ ( ) { } = ! < > | :
```

Esempio 1.

```
$keywords = "$40 for a g3/400";
$keywords = preg_quote ($keywords, "/");
echo $keywords; // returns \$40 for a g3\/400
```

Esempio 2. Esempio di come rendere in corsivo una qualsiasi parola di un testo

```
// In questo esempio, la funzione preg_quote($word) viene usata
// per impedire agli asterischi di avere il loro significato
// speciale per le espressioni regolari.

$testo = "Questo libro è *molto* difficile da trovare.";
$parola = "*molto*";
$testo = preg_replace ("/".preg_quote($parola)."/",
                          "<i>".$parola."</i>",
                          $testo);
```

preg_replace (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Esegue una ricerca ed una sostituzione con le espressioni regolari

mixed **preg_replace** (mixed espressione_regolare, mixed sostituto, mixed testo [, int limite]) \linebreak

La funzione ricerca in *testo* i criteri impostati in *espressione_regolare*. Se riconosce dei testi, li sostituisce con *sostituto*. Se si specifica *limite*, verranno sostituiti solamente *limite* testi riconosciuti; se viene omesso, oppure impostato a -1, verranno sostituiti tutti i testi riconosciuti.

Il parametro *sostituto* può contenere riferimenti nella forma di `\\n`, oppure, a partire dalla versione 4.0.4 di PHP, `$n` , con la preferenza per la seconda sintassi. Questo tipo di riferimento verrà sostituito dal testo che soddisfa l'*n* -esimo criterio presente in *espressione_regolare* . Sono ammessi numeri compresi tra 0 e 99 inclusi. Il valore 0 (`\\0` oppure `$0`) si riferisce al testo riconosciuto tramite tutta l'espressione regolare passata. Nel conteggio dei criteri di riconoscimento presenti, sono contate le parentesi aperte da sinistra verso destra partendo da 1.

Se verranno riconosciuti dei testi, la funzione restituisce la nuova versione di *testo*, altrimenti il parametro sarà restituito inalterato.

Ogni parametro passato alla funzione **preg_replace()** può essere una matrice.

Se il campo *testo* è una matrice, la ricerca e la sostituzione sarà eseguita su ogni elemento della matrice, e conseguentemente la funzione restituirà una matrice.

Se *espressione_regolare* ed *sostituto* sono entrambi delle matrici, la funzione **preg_replace()** utilizza gli elementi di ciascuna matrice per la ricerca e la sostituzione delle stringhe in *testo*. Nel caso in cui la matrice passata in *sostituto* abbia meno elementi che *espressione_regolare*, la funzione utilizzerà una stringa vuota per compensare gli elementi mancanti nella fase di sostituzione. Se, invece, *espressione_regolare* è una matrice, mentre il campo *sostituto* è una stringa, quest'ultima sarà usata come valore da sostituire ad ogni testo riconosciuto. Un discorso contrario non ha senso.

Il modificatore `/e`, permette alla funzione di considerare il testo posto in *sostituto* come codice PHP dopo aver valorizzato gli opportuni riferimenti. Suggerimento: accertarsi che *sostituto* sia del codice PHP valido, altrimenti si ottiene un errore di parsing alla linea della funzione **preg_replace()**.

Esempio 1. Esempi di sostituzione di valori

```
$patterns = array ("/(19|20)(\d{2})-(\d{1,2})-(\d{1,2})/",
                   "/^\s*{(\w+)}\s*=/");
$replace = array ("\\3/\\4/\\1\\2", "$\\1 =");
print preg_replace ($patterns, $replace, "{startDate} = 1999-5-27");
```

Questo esempio visualizzerà:

```
$startDate = 5/27/1999
```

Esempio 2. Utilizzo del modificatore /e

```
preg_replace ("/(<\/?)(\w+)([^>]*>)/e",
              "'\\1'.strtoupper('\\2').'\\3'",
              $html_body);
```

Nell’esempio precedente tutti i tag HTML presenti nel testo di input saranno convertiti in maiuscolo.

Esempio 3. Esempio di conversione di codice HTML in testo

```
// $document contiene un documento HTML.
// In questo esempio la funzione rimuove i tag HTML,
// le sezioni javascript e gli spazi bianchi.
// Inoltre si convertirà le entità HTML nella loro
// rappresentazione testuale.
$search = array ("’<script[^>]*?>.*?</script>’si",  // Rimozione del javascript
                 "’<[\/\!]*?[^<>]*?>’si",           // Rimozione dei tag HTML
                 "’([\r\n])[\s]+’",                 // Rimozione degli spazi bianchi
                 "’&(quot|#34);’i",                 // Sostituzione delle en-
tità HTML
                 "’&(amp|#38);’i",
                 "’&(lt|#60);’i",
                 "’&(gt|#62);’i",
                 "’&(nbsp|#160);’i",
                 "’&(iexcl|#161);’i",
                 "’&(cent|#162);’i",
                 "’&(pound|#163);’i",
                 "’&(copy|#169);’i",
                 "’&#(\d+);’e");                    // Valuta come codice PHP

$replace = array ("",
                  "",
                  "\\1",
                  "\"",
                  "&",
                  "<",
                  ">",
                  " ",
                  chr(161),
                  chr(162),
                  chr(163),
                  chr(169),
                  "chr(\\1)");

$text = preg_replace ($search, $replace, $document);
```

Nota: Il parametro `limite` è stato aggiunto successivamente alla versione 4.0.1pl2 di PHP.

Vedere anche preg_match(), preg_match_all() e preg_split().

preg_replace_callback (PHP 4 >= 4.0.5)

Esegue ricerche e sostituzioni con espressioni regolari usando il callback

mixed **preg_replace_callback** (mixed espressione_regolare, mixed callback, mixed testo [, int limite]) \linebreak

Fondamentalmente questa funzione si comporta come preg_replace(), eccetto che per la presenza del *callback*. Con quest'ultimo parametro si indica una funzione da richiamare a cui verrà passata una matrice con i testi riconosciuti in *testo*. La funzione di callback dovrebbe restituire la stringa da sostituire. Questa funzione è stata aggiunta nella versione 4.0.5 di PHP.

Vedere anche preg_replace().

preg_split (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Suddivisione di una stringa tramite le espressioni regolari

array **preg_split** (string espressione_regolare, string testo [, int limite [, int flags]]) \linebreak

> **Nota:** Il parametro *flags* è stato aggiunto nella versione 4 Beta 3 di PHP.

La funzione restituisce una matrice di parti di *testo* suddivisi tramite i criteri indicati da *espressione_regolare*.

Se viene specificato il parametro *limite*, la funzione restituisce tante parti del testo iniziale quante sono indicate da *limite*. Può essere usato il valore -1 per indicare "nessun limite". Ciò torna utile in abbinamento all'uso del parametro *flags*.

Il parametro *flags* può essere la combinazione dei seguenti flag (la combinazione di più flag avviene con l'operatore |):

PREG_SPLIT_NO_EMPTY

Specificando questo flag, la funzione **preg_split()** restituisce spezzoni di testo non vuoti.

PREG_SPLIT_DELIM_CAPTURE

Con l'uso di questo flag, la funzione cattura e restituisce eventuali espressioni poste tra parentesi nel parametro *espressione_regolare*. Questo flag è stato aggiunto nella versione 4.0.5.

Esempio 1. Esempio di preg_split(): Come ottenere le parti di un testo.

```
// Suddivide la seguente frase in base alla presenza di virgole, spazi bianchi,
// e altri caratteri speciali quali \r, \t, \n ed \f
$keywords = preg_split ("/[\s,]+/", "hypertext language, programming");
```

Esempio 2. Esempio di suddivisione di un testo in caratteri.

```
$str = 'string';
$chars = preg_split('//', $str, -1, PREG_SPLIT_NO_EMPTY);
print_r($chars);
```

Vedere anche spliti(), split(), implode(), preg_match(), preg_match_all() e preg_replace().

LXXXVII. Funzioni qtdom

qdom_error (PHP 4 >= 4.0.5)

Restituisce la stringa d'errore dell'ultima operazione QDOM o FALSE se non ci sono stati errori

string **qdom_error** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

qdom_tree (PHP 4 >= 4.0.4)

Crea un tree di una stringa xml

object **qdom_tree** (string) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

LXXXVIII. Funzioni per le espressioni regolari (POSIX estesa)

Nota: Il PHP, utilizzando le funzioni PCRE, supporta anche le espressioni regolari con una sintassi compatibile con Perl. Queste funzioni supportano riconoscimenti "pigliatutto", asserzioni, criteri condizionali, e diverse altre caratteristiche che non sono supportate dalla sintassi POSIX estesa.

Attenzione

Queste funzioni per l'espressioni regolari non sono binary-safe. Le funzioni PCRE lo sono.

In PHP, le espressioni regolari sono utilizzate per complesse manipolazioni di stringhe. Le funzioni che supportano le espressioni regolari sono:

- ereg()
- ereg_replace()
- eregi()
- eregi_replace()
- split()
- spliti()

Tutte queste funzioni usano una espressione regolare come loro primo argomento. Le espressioni regolari utilizzate da PHP sono di tipo POSIX esteso così come definito in POSIX 1003.2. Per una descrizione completa delle espressione regolari POSIX, vedere la pagina del manuale di regex inclusa nella directory di regex nella distribuzione di PHP. Questa è in formato man, pertanto per poterle leggere occorre eseguire **man /usr/local/src/regex/regex.7**.

Esempio 1. Esempi di espressione regolare

```
ereg ("abc", $string);
/* Restituisce vero se "abc"
   viene trovata ovunque in $string. */

ereg ("^abc", $string);
/* Restituisce vero se "abc"
   viene trovata all'inizio di $string. */

ereg ("abc$", $string);
/* Restituisce vero se "abc"
   viene trovata alla fine di $string. */

eregi ("(ozilla.[23]|MSIE.3)", $HTTP_USER_AGENT);
/* Restituisce vero se il browser
   è Netscape 2, 3 oppure MSIE 3. */
```

```
ereg ("([[:alnum:]]+) ([[:alnum:]]+) ([[:alnum:]]+)", $string,$regs);
/* Posizione tre parole separate da spazio
   in $regs[1], $regs[2] e $regs[3]. */

$string = ereg_replace ("^", "<br />", $string);
/* Posiziona il tag <br /> all’inizio di $string. */

$string = ereg_replace ("$", "<br />", $string);
/* Posiziona il tag <br /> alla fine di $string. */

$string = ereg_replace ("\n", "", $string);
/* Toglie ogni carattere di invio
   da $string. */
```

ereg (PHP 3, PHP 4 >= 4.0.0)

Riconoscimento di espressione regolare

int **ereg** (string epressione_regolare, string stringa [, array regs]) \linebreak

> **Nota:** Poichè utilizza espressioni regolari con sintassi compatibile con PERL, preg_match(), è spesso una alternativa più veloce a **ereg()**.

Ricerca in *stringa* testi che possano incrociarsi con l'espressione regolare indicata in *espressione_regolare*.

Se le parti di testo poste tra parentesi nel campo *espressione_regolare* sono incontrate nella *stringa* e la funzione viene chiamata utilizzando il terzo parametro *regs*, il testo riconosciuto sarà memorizzato nella matrice *regs*. L'indice 1, $regs[1], conterrà la sottostringa che parte dalla prima parentesi sinistra; $regs[2] conterrà la sottostringa a partire dalla seconda e così via. L'indice 0, $regs[0], conterrà la copia completa della stringa riconosciuta.

> **Nota:** Fino alla versione di PHP 4.1.0 compresa, `$regs` conterrà esattamente 10 elementi, anche se il numero delle stringhe riconosciute sia maggiore o minore di 10. Ciò non limita **ereg()** nella ricerca di più sottostringhe. Se non si riconoscono testi, `$regs` non sarà modificato da **ereg()**.

La ricerca è sensibile alle lettere maiuscole e minuscole.

La funzione ritorna `TRUE` se le ricerche previste da *espressione_regolare* sono riscontrate in *stringa*. Viene restituito `FALSE` se non si hanno riscontri, oppure si verificano degli errori.

Nel seguente frammento di codice, una data in formato ISO (YYYY-MM-DD) verrà visualizzata nel formato DD.MM.YYYY:

Esempio 1. ereg() Esempio

```
if (ereg ("([0-9]{4})-([0-9]{1,2})-([0-9]{1,2})", $data, $regs)) {
    echo "$regs[3].$regs[2].$regs[1]";
} else {
    echo "Formato di data non valido: $data";
}
```

Vedere anche eregi(), ereg_replace(), eregi_replace() e preg_match().

ereg_replace (PHP 3, PHP 4 >= 4.0.0)

Sostituzioni con espressioni regolari

string **ereg_replace** (string espressione_regolare, string testo_sostitutivo, string stringa) \linebreak

> **Nota:** Poichè utilizza espressioni regolari con sintassi compatibile con Perl, preg_replace(), è spesso una alternativa più veloce a **ereg_replace()**.

La funzione ricerca all'interno del parametro `stringa` sottostringhe che incrocino con le condizioni specificate in `espressione_regolare`. Quando sono trovate, queste vengono sostituite con il testo specificato in `testo_sostitutivo`.

La funzione restituisce la stringa modificata. (Ciò implica che se non ci sono sottostringhe che soddisfino l' espressione regolare, la funzione restituisce la stringa originaria).

Se in `espressione_regolare` si specificano delle sottostringhe utilizzando le parentesi, anche nel campo `testo_sostitutivo` si possono specificare delle sottostringhe di formato `\\digit`, che saranno sostituite dalle stringhe soddisfacenti la digit'esima condizione posta tra parentesi; `\\0` indica l'intera stringa. La funzione prevede che si possano utlizzare fino a nove sottostringhe. E' previsto che le parentesi siano nidificate, in questo caso il conteggio si basa sulle parentesi aperte.

Se nessuna parte di `stringa` viene riconosciuta, il parametro viene restituito invariato.

Come esempio il seguente frammento di codice visualizzerà la frase "Questo fu un test" tre volte:

Esempio 1. ereg_replace() Esempio

```
$stringa = "Questo è un test";
echo ereg_replace (" è", " fu", $stringa);
echo ereg_replace ("( )è", "\\1fu", $stringa);
echo ereg_replace ("(( )è)", "\\2fu", $stringa);
```

Un aspetto a cui occorre prestare attenzione riguarda l'uso di numero intero per il parametro `testo_sostitutivo`, si potrebbero ottenere risultati diversi da quanto atteso. Questo accade perchè la funzione **ereg_replace()** interpreta il numero come posizione ordinale di un carattere comportandosi di conseguenza. Ad esempio:

Esempio 2. ereg_replace() Esempio

```
<?php
/* Questo non si comporta come atteso. */
$num = 4;
$stringa = "Questa stringa ha quattro parole.";
$stringa = ereg_replace('quattro', $num, $stringa);
echo $stringa;   /* Risultato: 'Questa stringa ha   parole.' */

/* Questo funziona. */
$num = '4';
$stringa = "Questa stringa ha quattro parole.";
$stringa = ereg_replace('quattro', $num, $stringa);
echo $stringa;   /* Risultato: 'Questa stringa ha 4 parole' */
?>
```

Esempio 3. Sostituzione di URL

```
$testo = ereg_replace("[[:alpha:]]+://[^<>[:space:]]+[[:alnum:]/]",
                     "<a href=\"\\0\">\\0</a>", $testo);
```

Vedere anche ereg(), eregi(), eregi_replace(), str_replace() e preg_match().

eregi (PHP 3, PHP 4 >= 4.0.0)

Riconoscimento di espressioni regolari senza distinzione tra maiuscole e minuscole

int **eregi** (string espressione_regolare, string stringa [, array regs]) \linebreak

Questa funzione è identica a ereg(), tranne che per il fatto che non distingue tra lettere maiuscole e lettere minuscole.

Esempio 1. eregi() esempio

```
if (eregi("z", $stringa)) {
    echo "'$stringa' contiene una 'z' oppure una 'Z'!";
}
```

Vedere anche ereg(), ereg_replace(), e eregi_replace().

eregi_replace (PHP 3, PHP 4 >= 4.0.0)

Sostituzioni con espressioni regolari senza distinzione tra maiuscole e minuscole

string **eregi_replace** (string espressione_regolare, string testo_sostitutivo, string stringa) \linebreak

Questa funzione è identica a ereg_replace(), tranne che per il fatto che non distingue tra lettere maiuscole e lettere minuscole.

Vedere anche ereg(), eregi() e ereg_replace().

split (PHP 3, PHP 4 >= 4.0.0)

Suddivide una stringa in una matrice utilizzando le espressioni regolari

array **split** (string espressione_regolare, string stringa [, int limite]) \linebreak

> **Nota:** Poichè utilizza espressioni regolari con sintassi compatibile con Perl, preg_split(), è spesso una alternativa più veloce a **split()**.

La funzione restituisce una matrice di stringhe che sono delle sottostringhe del parametro *stringa*. Queste sono ottenute suddividendo il parametro secondo i limiti indicati dal parametro *espressione_regolare*. Se viene specificato il parametro *limite*, la funzione restituisce una matrice con un numero di elementi al massimo pari a *limite*. L'ultimo elemento della matrice contiene la parte restante del parametro *stringa* fornito. Se si verificano errori la funzione **split()** restituisce `FALSE`.

Esempio di come estrapolare i primi 4 campi da una linea del file `/etc/passwd`:

Esempio 1. Esempio di split()

```
list($user,$pass,$uid,$gid,$extra)= split (":", $passwd_line, 5);
```

> **Nota:** Se nella stringa passata vi sono *n* occorrenze del parametro *espressione_regolare*, la matrice restituita conterrà *n*+1 elementi. Invece, nel caso in cui non vi siano occorrenze della *espressione_regolare*, la matrice restituita conterrà un solo elemento. Ovviamente questo è valido anche nel caso in cui il parametro *stringa* è vuoto.

Nell'esempio che segue, si vedrà come analizzare una data il cui testo può contenere barre, punti o trattini:

Esempio 2. split() Esempio

```
$data = "04/30/1973";  // Delimitatori di testo: barre, punti, trattini
list ($mese, $giorno, $anno) = split ('[/.-]', $data);
echo "Mese: $mese; Giorno: $giorno; Anno: $anno<br>\n";
```

Fare attenzione al fatto che *espressione_regolare* è distingue tra lettere maiuscole e minuscole.

Nota: se non è richiesta la potenza delle espressioni regolari, è più veloce la funzione explode(), la quale non richiede l'uso del motore delle espressioni regolari.

Gli utenti che cercano un modo di emulare il comportamento di Perl **@chars = split('', $str)**, sono rimandati agli esempi di preg_split().

Occorre fare attenzione al fatto che il parametro *espressione_regolare* è una espressione regolare e, pertanto, se si devono riconoscere caratteri che sono considerati speciali per le espressioni regolari, occorre codificarli con i caratteri di escape. Se si ritiene che la funzione **split ()** (o anche le altre funzioni derivate da regex) si comportino in modo anomalo, è opportuno leggere il file `regex.7`, incluso nella cartella `regex/` della distribuzione di PHP. Questo file è nel formato del

manuale di unix (man), pertanto per visualizzarlo occorre eseguire il comando **man /usr/local/src/regex/regex.7**.

Vedere anche: preg_split(), spliti(), explode(), implode(), chunk_split() e wordwrap().

spliti (PHP 4)

Suddivide una stringa in una matrice usando le espressioni regolari senza distinguere tra maiuscole e minuscole

array **spliti** (string espressione_regolare, string stringa [, int limite]) \linebreak

Questa funzione ha un comportamento identico a split() tranne che per il fatto di non distinguere tra lettere maiuscole e minuscole.

Vedere anche **preg_spliti()**, split(), explode() e implode().

sql_regcase (PHP 3, PHP 4 >= 4.0.0)

Genera una espressione regolare per riconoscimenti senza distinguere tra maiuscole e minuscole

string **sql_regcase** (string stringa) \linebreak

La funzione restituisce una espressione regolare che sia in grado di riconoscere il parametro *`stringa`*, a prescindere dalle lettere maiuscole minuscole. L'espressione regolare restituita corrisponde a *`string`* con ciascun carattere riportato tra parentesi. Ogni parentesi contiene il singolo carattere in maiuscolo ed in minuscolo. Se il carattere non esiste in forma minuscola o maiuscola, il carattere originale vie riportato due volte.

Esempio 1. sql_regcase() Esempio

```
echo sql_regcase ("Foo bar");
```

visualizza

```
[Ff][Oo][Oo] [Bb][Aa][Rr]
```

.

Questa funzione torna utile quando si devono ottenere espressioni regolari non distinguono tra lettere maiuscole e minuscole da passare a prodotti che supportano espressioni regolari che distinguono il tipo di lettera.

LXXXIX. Funzioni per i semafori, la memoria condivisa ed IPC

Questo modulo fornisce le funzioni relative all'IPC di System V. Queste includono semafori, memoria condivisa e messaggi tra i processi (IPC).

I semafori possono essere utilizzati per fornire un accesso esclusivo alle risorse sulla macchina corrente, oppure per limitare il numero di processi che possono utilizzare simultaneamente una risorsa.

Questo modulo fornisce anche le funzioni per la memoria condivisa a partire dalla gestione della memoria condivisa di System V. La memoria condivisa può essere utilizzata per fornire l'accesso a variabili globali. Differenti demoni httpd e anche altri programmi (tipo Perl, C, ...) sono in grado di accedere a questi dati creando uno scambio di dati globale. Si ricordi che la memoria condivisa non è garantita nei confronti di accessi simultanei. Si utilizzino i semafori per la sincronizzazione.

Tabella 1. Limiti della memoria condivisa posti da UNIX

SHMMAX	dimensione massima della memoria condivisa, solitamente 131072 bytes
SHMMIN	dimensione minima della memoria condivisa, solitamente 1 byte
SHMMNI	massimo ammontare dei segmenti di memoria condivisa sul sistema, solitamente 100
SHMSEG	numero massimo di segmenti di memoria condivisa per processo, solitamente 6

Le funzioni relative ai messaggi possono essere usate per inviare e ricevere messaggi da/per altri processi. Esse permetto un semplice ed efficace metodo di interscambio dati tra i processi, senza dovere ricorrere ad alternative quali i socket nel dominio unix.

Nota: Queste funzioni non sono attive sui sistemi Windows.

ftok (PHP 4 >= 4.2.0)

Converte il percorso e un identificatore di progetto in un chiave IPC di System V

int **ftok** (string pathname, string proj) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

msg_get_queue (unknown)

Create or attach to a message queue

int **msg_get_queue** (int key [, int perms]) \linebreak

msg_get_queue() returns an id that can be used to access the System V message queue with the given *key*. The first call creates the message queue with the optional *perms* (default: 0666). A second call to **msg_get_queue()** for the same *key* will return a different message queue identifier, but both identifiers access the same underlying message queue. If the message queue already exists, the *perms* will be ignored.

See also: msg_remove_queue(), msg_receive(), msg_send(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_receive (unknown)

Receive a message from a message queue

bool **msg_receive** (int queue, int desiredmsgtype, int msgtype, int maxsize, mixed message [, bool unserialize [, int flags [, int errorcode]]]) \linebreak

msg_receive() will receive the first message from the specified *queue* of the type specified by *desiredmsgtype*. The type of the message that was received will be stored in *msgtype*. The maximum size of message to be accepted is specified by the *maxsize*; if the message in the queue is larger than this size the function will fail (unless you set *flags* as described below). The received message will be stored in *message*, unless there were errors receiving the message, in which case the optional *errorcode* will be set to the value of the system errno variable to help you identify the cause.

If *desiredmsgtype* is 0, the message from the front of the queue is returned. If *desiredmsgtype* is greater than 0, then the first message of that type is returned. If *desiredmsgtype* is less than 0, the first message on the queue with the lowest type less than or equal to the absolute value of *desiredmsgtype* will be read. If no messages match the criteria,

your script will wait until a suitable message arrives on the queue. You can prevent the script from blocking by specifying MSG_IPC_NOWAIT in the *flags* parameter.

unserialize defaults to TRUE; if it is set to TRUE, the message is treated as though it was serialized using the same mechanism as the session module. The message will be unserialized and then returned to your script. This allows you to easily receive arrays or complex object structures from other PHP scripts, or if you are using the WDDX serializer, from any WDDX compatible source. If *unserialize* is FALSE, the message will be returned as a binary-safe string.

The optional *flags* allows you to pass flags to the low-level msgrcv system call. It defaults to 0, but you may specify one or more of the following values (by adding or ORing them together).

Tabella 1. Flag values for msg_receive

MSG_IPC_NOWAIT	If there are no messages of the *desiredmsgtype*, return immediately and do not wait. The function will fail and return an integer value corresponding to ENOMSG.
MSG_EXCEPT	Using this flag in combination with a *desiredmsgtype* greater than 0 will cause the function to receive the first message that is not equal to *desiredmsgtype*.
MSG_NOERROR	If the message is longer than *maxsize*, setting this flag will truncate the message to *maxsize* and will not signal an error.

Upon successful completion the message queue data structure is updated as follows: *msg_lrpid* is set to the process-ID of the calling process, *msg_qnum* is decremented by 1 and *msg_rtime* is set to the current time.

msg_receive() returns TRUE on success or FALSE on failure. If the function fails, the optional *errorcode* will be set to the value of the system errno variable.

See also: msg_remove_queue(), msg_send(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_remove_queue (unknown)

Destroy a message queue

bool **msg_remove_queue** (int queue) \linebreak

msg_remove_queue() destroys the message queue specified by the *queue*. Only use this function when all processes have finished working with the message queue and you need to release the system resources held by it.

See also: **msg_remove_queue()**, msg_receive(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_send (unknown)

Send a message to a message queue

bool **msg_send** (int queue, int msgtype, mixed message [, bool serialize [, bool blocking [, int errorcode]]]) \linebreak

msg_send() sends a *message* of type *msgtype* (which MUST be greater than 0) to a the message queue specified by *queue*.

If the message is too large to fit in the queue, your script will wait until another process reads messages from the queue and frees enough space for your message to be sent. This is called blocking; you can prevent blocking by setting the optional *blocking* parameter to FALSE, in which case **msg_send()** will immediately return FALSE if the message is too big for the queue, and set the optional *errorcode* to EAGAIN, indicating that you should try to send your message again a little later on.

The optional *serialize* controls how the *message* is sent. *serialize* defaults to TRUE which means that the *message* is serialized using the same mechanism as the session module before being sent to the queue. This allows complex arrays and objects to be sent to other PHP scripts, or if you are using the WDDX serializer, to any WDDX compatible client.

Upon successful completion the message queue data structure is updated as follows: *msg_lspid* is set to the process-ID of the calling process, *msg_qnum* is incremented by 1 and *msg_stime* is set to the current time.

See also: msg_remove_queue(), msg_receive(), msg_stat_queue() and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

msg_set_queue (unknown)

Set information in the message queue data structure

bool **msg_set_queue** (int queue, array data) \linebreak

msg_set_queue() allows you to change the values of the msg_perm.uid, msg_perm.gid, msg_perm.mode and msg_qbytes fields of the underlying message queue data structure. You specify the values you require by setting the value of the keys that you require in the *data* array.

Changing the data structure will require that PHP be running as the same user that created the the queue, owns the queue (as determined by the existing msg_perm.xxx fields), or be running with root privileges. root privileges are required to raise the msg_qbytes values above the system defined limit.

See also: msg_remove_queue(), msg_receive(), msg_stat_queue() and **msg_set_queue()**.

This function was introduced in PHP 4.3.0 (not yet released).

msg_stat_queue (unknown)

Returns information from the message queue data structure

array **msg_stat_queue** (int queue) \linebreak

msg_stat_queue() returns the message queue meta data for the message queue specified by the *queue*. This is useful, for example, to determine which process sent the message that was just received.

The return value is an array whose keys and values have the following meanings:

Tabella 1. Array structure for msg_stat_queue

msg_perm.uid	The uid of the owner of the queue.
msg_perm.gid	The gid of the owner of the queue.
msg_perm.mode	The file access mode of the queue.
msg_stime	The time that the last message was sent to the queue.
msg_rtime	The time that the last message was received from the queue.
msg_ctime	The time that the queue was last changed.
msg_qnum	The number of messages waiting to be read from the queue.
msg_qbytes	The number of bytes of space currently available in the queue to hold sent messages until they are received.
msg_lspid	The pid of the process that sent the last message to the queue.
msg_lrpid	The pid of the process that received the last message from the queue.

See also: msg_remove_queue(), msg_receive(), **msg_stat_queue()** and msg_set_queue().

This function was introduced in PHP 4.3.0 (not yet released).

sem_acquire (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Acquisisce un semaforo

bool **sem_acquire** (int sem_identifier) \linebreak

La funzione restituisce: `TRUE` se ha successo, `FALSE` se si verifica un errore.

La funzione **sem_acquire()** si blocca (se necessario) fino a quando non riesce ad acquisire il semaforo. Se un processo tenta di acquisire un semaforo che ha già acquisito può restare bloccato per sempre se la nuova acquisizione del semaforo causa il superamento del parametro max_acquire.

Dopo avere processato una richiesta, qualsiasi semaforo acquisito dal processo, ma non esplicitamente rilasciato, sarà rilasciato automaticamente e causerà un messaggio di warning.

Vedere anche: sem_get() e sem_release().

sem_get (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Ottiene l'id di un semaforo

int **sem_get** (int key [, int max_acquire [, int perm]]) \linebreak

La funzione ritorna un identificatore positivo di semaforo se ha successo, oppure `FALSE` se si verifica un errore.

La funzione **sem_get()** restituisce un identificativo che può essere utilizzato per accedere al semaforo con chiave indicata in *`key`*. Se necessario il semaforo viene creato con i bit dei permessi valorizzati come specificato in perm (di default 666). In max_acquire si indicat il numero massimo di processi che possono acquisire il semaforo simultaneamente (1 per default). In realtà questo valore è modificabile solo se il processo è l'unico, in quel momento, ad essere collegato al semaforo.

Una seconds chiamata a **sem_get()** per la medesima chiave restituisce un identificativo di semaforo differente, ma entrambi gli gli identificativi accedono al medesimo semaforo sottostante.

Vedere anche: sem_acquire(), sem_release() e ftok().

> **Nota:** Questa funzione non è utilizzabile sui sistemi Windows.

sem_release (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Rilascia un semaforo

bool **sem_release** (int sem_identifier) \linebreak

La funzione restituisce: `TRUE` se ha successo, `FALSE` se si verifica un errore.

La funzione **sem_release()** rilascia il semaforo se questo è attualmente acquisito dal processo chiamante. In caso contrario sarà generato un messaggio di warning.

Dopo avere rilasciato un semaforo, occorre eseguire di nuovo sem_acquire() per ri-acquisirlo.

Vedere anche: sem_get() e sem_acquire().

> **Nota:** Questa funzione non è utilizzabile sui sistemi Windows.

sem_remove (PHP 4 >= 4.1.0)

Rimuove un semaforo

bool **sem_remove** (int sem_identifier) \linebreak

La funzione restituisce: `TRUE` se ha successo, `FALSE` se si verifica un errore.

La funzione **sem_remove()** rimuove il semaforo indicato da *`sem_identifier`* se questo è stato generato in orecedenza da sem_get(). In caso contrario si genera un messaggio di warning

Una volta rimosso, il semaforo non è più accessibile.

Vedere anche: sem_get(), sem_release() e sem_acquire().

> **Nota:** Questa funzione non è utilizzabile sui sistemi Windows. Questa funzione è stata aggiunta nella versione 4.1.0 di PHP.

shm_attach (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Crea oppure apre un segmento di memoria condivisa

int **shm_attach** (int key [, int memsize [, int perm]]) \linebreak

La funzione **shm_attach()** restituisce un identificativo che può essere utilizzato per accedere alla memoria condivisa identificata dalla chiave *key* la prima chiamata crea il segmento di memoria condivisa (la dimensione di default può essere indicata in sysvshm.init_mem nel file di configurazione, altrimenti viene fissata a 10000 bytes) e con i bit dei permessi (default: 0666).

Una seconda chiamata alla funzione **shm_attach()** con il medesimo parametro *key* restituirà un identificativo di memoria condivisa differente, ma entrambi accederanno alla medesima memoria condivisa sottostante. I parametri *Memsize* e *perm* saranno ignorati.

Vedere anche: ftok().

> **Nota:** Questa funzione non è utilizzabile sui sistemi Windows.

shm_detach (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Disconnette da un segmento di memoria condivisa

int **shm_detach** (int shm_identifier) \linebreak

La funzione **shm_detach()** disconnette dal segmento di memoria condivisa indicato dal parametro *shm_identifier* creato tramite la funzione shm_attach(). Si ricordi che la memoria condivisa continua a esistere nel sistema Unix e i dati sono ancora presenti.

shm_get_var (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce una variabile dalla memoria condivisa

mixed **shm_get_var** (int id, int variable_key) \linebreak

La funzione **shm_get_var()** restituisce la variabile identificata dalla chiave *variable_key*. La variabile resta presente nella memoria condivisa.

Nota: Questa funzione non è utilizzabile sui sistemi Windows.

shm_put_var (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Inserisce o aggiorna una variabile nella memoria condivisa

int **shm_put_var** (int shm_identifier, int variable_key, mixed variable) \linebreak

La funzione inserisce o aggiorna la variabile indicata in *`variable`* con chiave *`variable_key`*. Sono suportati Tutti i tipi di variabili.

Nota: Questa funzione non è utilizzabile sui sistemi Windows.

shm_remove (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Rimuove un segmento di memoria condivisa dal sistema Unix

int **shm_remove** (int shm_identifier) \linebreak

La funzione rimuove un segmento di memoria condivisa dal sistsema. Tutti i dati contenuti saranno persi.

Nota: Questa funzione non è utilizzabile sui sistemi Windows.

shm_remove_var (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Rimuove una variabile dalla memoria condivisa

int **shm_remove_var** (int id, int variable_key) \linebreak

Rimuove la variabile identificata da *`variable_key`* e libera la memoria occupata.

Nota: Questa funzione non è utilizzabile sui sistemi Windows.

XC. SESAM database functions

SESAM/SQL-Server is a mainframe database system, developed by Fujitsu Siemens Computers, Germany. It runs on high-end mainframe servers using the operating system BS2000/OSD.

In numerous productive BS2000 installations, SESAM/SQL-Server has proven ...

- the ease of use of Java-, Web- and client/server connectivity,
- the capability to work with an availability of more than 99.99%,
- the ability to manage tens and even hundreds of thousands of users.

Now there is a PHP3 SESAM interface available which allows database operations via PHP-scripts.

Configuration notes: There is no standalone support for the PHP SESAM interface, it works only as an integrated Apache module. In the Apache PHP module, this SESAM interface is configured using Apache directives.

Tabella 1. SESAM Configuration directives

Directive	Meaning
`php3_sesam_oml`	Name of BS2000 PLAM library containing the loadable SESAM driver modules. Required for using SESAM functions. Example: `php3_sesam_oml $.SYSLNK.SESAM-SQL.030`

`php3_sesam_configfile`	Name of SESAM application configuration file. Required for using SESAM functions. Example:
	
	`php3_sesam_configfile $SESAM.SESAM.CONF.AW`
	
	It will usually contain a configuration like (see SESAM reference manual):
	
	`CNF=B`
	`NAM=K`
	`NOTYPE`

`php3_sesam_messagecatalog`	Name of SESAM message catalog file. In most cases, this directive is not neccessary. Only if the SESAM message file is not installed in the system's BS2000 message file table, it can be set with this directive. Example: `php3_sesam_messagecatalog $.SYSMES.SESAM-SQL.030`

In addition to the configuration of the PHP/SESAM interface, you have to configure the SESAM-Database server itself on your mainframe as usual. That means:

- starting the SESAM database handler (DBH), and
- connecting the databases with the SESAM database handler

To get a connection between a PHP script and the database handler, the `CNF` and `NAM` parameters of the selected SESAM configuration file must match the id of the started database handler.

In case of distributed databases you have to start a SESAM/SQL-DCN agent with the distribution table including the host and database names.

The communication between PHP (running in the POSIX subsystem) and the database handler (running outside the POSIX subsystem) is realized by a special driver module called SQLSCI and SESAM connection modules using common memory. Because of the common memory access, and because PHP is a static part of the web server, database accesses are very fast, as they do not require remote accesses via ODBC, JDBC or UTM.

Only a small stub loader (SESMOD) is linked with PHP, and the SESAM connection modules are pulled in from SESAM's OML PLAM library. In the configuration, you must tell PHP the name of this PLAM library, and the file link to use for the SESAM configuration file (As of SESAM V3.0, SQLSCI is available in the SESAM Tool Library, which is part of the standard distribution).

Because the SQL command quoting for single quotes uses duplicated single quotes (as opposed to a single quote preceded by a backslash, used in some other databases), it is advisable to set the PHP configuration directives `php3_magic_quotes_gpc` and `php3_magic_quotes_sybase` to `On` for all PHP scripts using the SESAM interface.

Runtime considerations: Because of limitations of the BS2000 process model, the driver can be loaded only after the Apache server has forked off its server child processes. This will slightly slow down the initial SESAM request of each child, but subsequent accesses will respond at full speed.

When explicitly defining a Message Catalog for SESAM, that catalog will be loaded each time the driver is loaded (i.e., at the initial SESAM request). The BS2000 operating system prints a message after successful load of the message catalog, which will be sent to Apache's error_log file. BS2000 currently does not allow suppression of this message, it will slowly fill up the log.

Make sure that the SESAM OML PLAM library and SESAM configuration file are readable by the user id running the web server. Otherwise, the server will be unable to load the driver, and will not allow to call any SESAM functions. Also, access to the database must be granted to the user id under which the Apache server is running. Otherwise, connections to the SESAM database handler will fail.

Cursor Types: The result cursors which are allocated for SQL "select type" queries can be either "sequential" or "scrollable". Because of the larger memory overhead needed by "scrollable" cursors, the default is "sequential".

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: `SESAM_SEEK_NEXT`) and the scrolling offset which can either be set once by sesam_seek_row() or each time when fetching a row using sesam_fetch_row(). When fetching a row using a

"scrollable" cursor, the following post-processing is done for the global default values for the scrolling type and scrolling offset:

Tabella 2. Scrolled Cursor Post-Processing

Scroll Type	Action
`SESAM_SEEK_NEXT`	none
`SESAM_SEEK_PRIOR`	none
`SESAM_SEEK_FIRST`	set scroll type to `SESAM_SEEK_NEXT`
`SESAM_SEEK_LAST`	set scroll type to `SESAM_SEEK_PRIOR`
`SESAM_SEEK_ABSOLUTE`	Auto-Increment internal offset value
`SESAM_SEEK_RELATIVE`	none. (maintain global default *`offset`* value, which allows for, e.g., fetching each 10th row backwards)

Porting note: Because in the PHP world it is natural to start indexes at zero (rather than 1), some adaptions have been made to the SESAM interface: whenever an indexed array is starting with index 1 in the native SESAM interface, the PHP interface uses index 0 as a starting point. E.g., when retrieving columns with sesam_fetch_row(), the first column has the index 0, and the subsequent columns have indexes up to (but not including) the column count ($array["count"]). When porting SESAM applications from other high level languages to PHP, be aware of this changed interface. Where appropriate, the description of the respective php sesam functions include a note that the index is zero based.

Security concerns: When allowing access to the SESAM databases, the web server user should only have as little privileges as possible. For most databases, only read access privilege should be granted. Depending on your usage scenario, add more access rights as you see fit. Never allow full control to any database for any user from the 'net! Restrict access to php scripts which must administer the database by using password control and/or SSL security.

Migration from other SQL databases: No two SQL dialects are ever 100% compatible. When porting SQL applications from other database interfaces to SESAM, some adaption may be required. The following typical differences should be noted:

- Vendor specific data types

  Some vendor specific data types may have to be replaced by standard SQL data types (e.g., `TEXT` could be replaced by `VARCHAR(max. size)`).

- Keywords as SQL identifiers

  In SESAM (as in standard SQL), such identifiers must be enclosed in double quotes (or

renamed).

- Display length in data types

  SESAM data types have a precision, not a display length. Instead of `int(4)` (intended use: integers up to ’9999’), SESAM requires simply `int` for an implied size of 31 bits. Also, the only datetime data types available in SESAM are: `DATE`, `TIME(3)` and `TIMESTAMP(3)`.

- SQL types with vendor-specific `unsigned`, `zerofill`, or `auto_increment` attributes

  `Unsigned` and `zerofill` are not supported. `Auto_increment` is automatic (use `"INSERT ... VALUES(*, ...)"` instead of `"... VALUES(0, ...)"` to take advantage of SESAM-implied auto-increment.

- **int ... DEFAULT ’0000’**

  Numeric variables must not be initialized with string constants. Use **DEFAULT 0** instead. To initialize variables of the datetime SQL data types, the initialization string must be prefixed with the respective type keyword, as in: `CREATE TABLE exmpl ( xtime timestamp(3) DEFAULT TIMESTAMP ’1970-01-01 00:00:00.000’ NOT NULL );`

- **$count = xxxx_num_rows();**

  Some databases promise to guess/estimate the number of the rows in a query result, even though the returned value is grossly incorrect. SESAM does not know the number of rows in a query result before actually fetching them. If you REALLY need the count, try **SELECT COUNT(...) WHERE ...**, it will tell you the number of hits. A second query will (hopefully) return the results.

- **DROP TABLE thename;**

  In SESAM, in the **DROP TABLE** command, the table name must be either followed by the keyword `RESTRICT` or `CASCADE`. When specifying `RESTRICT`, an error is returned if there are dependent objects (e.g., VIEWs), while with `CASCADE`, dependent objects will be deleted along with the specified table.

Notes on the use of various SQL types: SESAM does not currently support the BLOB type. A future version of SESAM will have support for BLOB.

At the PHP interface, the following type conversions are automatically applied when retrieving SQL fields:

Tabella 3. SQL to PHP Type Conversions

SQL Type	PHP Type
SMALLINT, INTEGER	integer
NUMERIC, DECIMAL, FLOAT, REAL, DOUBLE	float
DATE, TIME, TIMESTAMP	string
VARCHAR, CHARACTER	string

When retrieving a complete row, the result is returned as an array. Empty fields are not filled in, so you will have to check for the existence of the individual fields yourself (use isset() or empty() to test for empty fields). That allows more user control over the appearance of empty fields (than in the case of an empty string as the representation of an empty field).

Support of SESAM's "multiple fields" feature: The special "multiple fields" feature of SESAM allows a column to consist of an array of fields. Such a "multiple field" column can be created like this:

Esempio 1. Creating a "multiple field" column

```
CREATE TABLE multi_field_test (
    pkey CHAR(20) PRIMARY KEY,
    multi(3) CHAR(12)
)
```

and can be filled in using:

Esempio 2. Filling a "multiple field" column

```
INSERT INTO multi_field_test (pkey, multi(2..3) )
    VALUES ('Second', <'first_val', 'second_val'>)
```

Note that (like in this case) leading empty sub-fields are ignored, and the filled-in values are collapsed, so that in the above example the result will appear as multi(1..2) instead of multi(2..3).

When retrieving a result row, "multiple columns" are accessed like "inlined" additional columns. In the example above, "pkey" will have the index 0, and the three "multi(1..3)" columns will be accessible as indices 1 through 3.

For specific SESAM details, please refer to the SESAM/SQL-Server documentation (english) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_en.htm) or the SESAM/SQL-Server documentation (german) (http://its.siemens.de/lobs/its/techinf/oltp/sesam/manuals/index_gr.htm), both available online, or use the respective manuals.

sesam_affected_rows (PHP 3 CVS only)

Get number of rows affected by an immediate query

int **sesam_affected_rows** (string result_id) \linebreak

`result_id` is a valid result id returned by sesam_query().

Returns the number of rows affected by a query associated with *`result_id`*.

The **sesam_affected_rows()** function can only return useful values when used in combination with "immediate" SQL statements (updating operations like `INSERT`, `UPDATE` and `DELETE`) because SESAM does not deliver any "affected rows" information for "select type" queries. The number returned is the number of affected rows.

See also: sesam_query() and sesam_execimm()

```
$result = sesam_execimm ("DELETE FROM PHONE WHERE LASTNAME = '".strtoupper ($name)."'");
if (!$result) {
    ... error ...
}
print sesam_affected_rows ($result).
    " entries with last name ".$name." deleted.\n"
```

sesam_commit (PHP 3 CVS only)

Commit pending updates to the SESAM database

bool **sesam_commit** (void) \linebreak

Returns: `TRUE` on success, `FALSE` on errors

sesam_commit() commits any pending updates to the database.

Note that there is no "auto-commit" feature as in other databases, as it could lead to accidental data loss. Uncommitted data at the end of the current script (or when calling sesam_disconnect()) will be discarded by an implied sesam_rollback() call.

See also: sesam_rollback().

Esempio 1. Committing an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (!sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)"))
        die("insert failed");
    if (!sesam_commit())
        die("commit failed");
}
?>
```

sesam_connect (PHP 3 CVS only)

Open SESAM database connection

bool **sesam_connect** (string catalog, string schema, string user) \linebreak

Returns `TRUE` if a connection to the SESAM database was made, or `FALSE` on error.

sesam_connect() establishes a connection to an SESAM database handler task. The connection is always "persistent" in the sense that only the very first invocation will actually load the driver from the configured SESAM OML PLAM library. Subsequent calls will reuse the driver and will immediately use the given catalog, schema, and user.

When creating a database, the *"catalog"* name is specified in the SESAM configuration directive **//ADD-SQL-DATABASE-CATALOG-LIST ENTRY-1 = *CATALOG(CATALOG-NAME = catalogname,...)**

The *"schema"* references the desired database schema (see SESAM handbook).

The *"user"* argument references one of the users which are allowed to access this *"catalog"* / *"schema"* combination. Note that *"user"* is completely independent from both the system's user id's and from HTTP user/password protection. It appears in the SESAM configuration only.

See also sesam_disconnect().

Esempio 1. Connect to a SESAM database

```
<?php
if (!sesam_connect ("mycatalog", "myschema", "otto")
    die("Unable to connect to SESAM");
?>
```

sesam_diagnostic (PHP 3 CVS only)

Return status information for last SESAM call

array **sesam_diagnostic** (void) \linebreak

Returns an associative array of status and return codes for the last SQL query/statement/command. Elements of the array are:

Element	Contents

Tabella 1. Status information returned by sesam_diagnostic()

Element	Contents
$array["sqlstate"]	5 digit SQL return code (see the SESAM manual for the description of the possible values of SQLSTATE)
$array["rowcount"]	number of affected rows in last update/insert/delete (set after "immediate" statements only)
$array["errmsg"]	"human readable" error message string (set after errors only)
$array["errcol"]	error column number of previous error (0-based; or -1 if undefined. Set after errors only)
$array["errlin"]	error line number of previous error (0-based; or -1 if undefined. Set after errors only)

In the following example, a syntax error (E SEW42AE ILLEGAL CHARACTER) is displayed by including the offending SQL statement and pointing to the error location:

Esempio 1. Displaying SESAM error messages with error position

```
<?php
// Function which prints a formatted error message,
// displaying a pointer to the syntax error in the
// SQL statement
function PrintReturncode ($exec_str) {
    $err = Sesam_Diagnostic();
    $colspan=4; // 4 cols for: sqlstate, errlin, errcol, rowcount
    if ($err["errlin"] == -1)
        --$colspan;
    if ($err["errcol"] == -1)
        --$colspan;
    if ($err["rowcount"] == 0)
        --$colspan;
    echo "<TABLE BORDER>\n";
    echo "<TR><TH COLSPAN=".$colspan."><FONT COLOR=red>ERROR:</FONT> ".
   htmlspecialchars($err["errmsg"])."</TH></TR>\n";
    if ($err["errcol"] >= 0) {
        echo "<TR><TD COLSPAN=".$colspan."><PRE>\n";
        $errstmt = $exec_str."\n";
        for ($lin=0; $errstmt != ""; ++$lin) {
            if ($lin != $err["errlin"]) { // $lin is less or greater than errlin
                if (!($i = strchr ($errstmt, "\n")))
                    $i = "";
                $line = substr ($errstmt, 0, strlen($errstmt)-strlen($i)+1);
                $errstmt = substr($i, 1);
                if ($line != "\n")
                    print htmlspecialchars ($line);
            } else {
                if (! ($i = strchr ($errstmt, "\n")))
```

```
                $i = "";
            $line = substr ($errstmt, 0, strlen ($errstmt)-strlen($i)+1);
            $errstmt = substr($i, 1);
            for ($col=0; $col < $err["errcol"]; ++$col)
                echo (substr($line, $col, 1) == "\t") ? "\t" : ".";
            echo "<FONT COLOR=RED><BLINK>\\</BLINK></FONT>\n";
            print "<FONT COLOR=\"#880000\">".htmlspecialchars($line)."</FONT>";
            for ($col=0; $col < $err["errcol"]; ++$col)
                echo (substr ($line, $col, 1) == "\t") ? "\t" : ".";
            echo "<FONT COLOR=RED><BLINK>/</BLINK></FONT>\n";
        }
    }
    echo "</PRE></TD></TR>\n";
    }
    echo "<TR>\n";
    echo " <TD>sqlstate=" . $err["sqlstate"] . "</TD>\n";
    if ($err["errlin"] != -1)
        echo " <TD>errlin=" . $err["errlin"] . "</TD>\n";
    if ($err["errcol"] != -1)
        echo " <TD>errcol=" . $err["errcol"] . "</TD>\n";
    if ($err["rowcount"] != 0)
         echo " <TD>rowcount=" . $err["rowcount"] . "</TD>\n";
    echo "</TR>\n";
    echo "</TABLE>\n";
}

if (!sesam_connect ("mycatalog", "phoneno", "otto"))
  die ("cannot connect");

$stmt = "SELECT * FROM phone\n".
        " WHERE@ LASTNAME='KRAEMER'\n".
        " ORDER BY FIRSTNAME";
if (!($result = sesam_query ($stmt)))
    PrintReturncode ($stmt);
?>
```

See also: sesam_errormsg() for simple access to the error string only

sesam_disconnect (PHP 3 CVS only)

Detach from SESAM connection

bool **sesam_disconnect** (void) \linebreak

Returns: always `TRUE`.

sesam_disconnect() closes the logical link to a SESAM database (without actually disconnecting and unloading the driver).

Note that this isn't usually necessary, as the open connection is automatically closed at the end of the script's execution. Uncommitted data will be discarded, because an implicit sesam_rollback() is executed.

sesam_disconnect() will not close the persistent link, it will only invalidate the currently defined *"catalog"*, *"schema"* and *"user"* triple, so that any sesam function called after **sesam_disconnect()** will fail.

See also: sesam_connect().

Esempio 1. Closing a SESAM connection

```
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    ... some queries and stuff ...
    sesam_disconnect();
}
```

sesam_errormsg (PHP 3 CVS only)

Returns error message of last SESAM call

string **sesam_errormsg** (void) \linebreak

Returns the SESAM error message associated with the most recent SESAM error.

```
if (!sesam_execimm ($stmt))
  printf ("%s<br>\n", sesam_errormsg());
```

See also: sesam_diagnostic() for the full set of SESAM SQL status information

sesam_execimm (PHP 3 CVS only)

Execute an "immediate" SQL-statement

string **sesam_execimm** (string query) \linebreak

Returns: A SESAM "result identifier" on success, or `FALSE` on error.

sesam_execimm() executes an "immediate" statement (i.e., a statement like UPDATE, INSERT or DELETE which returns no result, and has no INPUT or OUTPUT variables). "select type" queries can not be used with **sesam_execimm()**. Sets the *affected_rows* value for retrieval by the sesam_affected_rows() function.

Note that sesam_query() can handle both "immediate" and "select-type" queries. Use **sesam_execimm()** only if you know beforehand what type of statement will be executed. An attempt to use SELECT type queries with **sesam_execimm()** will return `$err["sqlstate"] == "42SBW"`.

The returned "result identifier" can not be used for retrieving anything but the sesam_affected_rows(); it is only returned for symmetry with the sesam_query() function.

```
$stmt = "INSERT INTO mytable VALUES ('one', 'two')";
$result = sesam_execimm ($stmt);
$err = sesam_diagnostic();
print ("sqlstate = ".$err["sqlstate"]."\n".
       "Affected rows = ".$err["rowcount"]." == ".
       sesam_affected_rows($result)."\n");
```

See also: sesam_query() and sesam_affected_rows().

sesam_fetch_array (PHP 3 CVS only)

Fetch one row as an associative array

array **sesam_fetch_array** (string result_id [, int whence [, int offset]]) \linebreak

Returns an array that corresponds to the fetched row, or `FALSE` if there are no more rows.

sesam_fetch_array() is an alternative version of sesam_fetch_row(). Instead of storing the data in the numeric indices of the result array, it stores the data in associative indices, using the field names as keys.

result_id is a valid result id returned by sesam_query() (select type queries only!).

For the valid values of the optional *whence*and *offset* parameters, see the sesam_fetch_row() function for details.

sesam_fetch_array() fetches one row of data from the result associated with the specified result identifier. The row is returned as an associative array. Each result column is stored with an associative index equal to its column (aka. field) name. The column names are converted to lower case.

Columns without a field name (e.g., results of arithmetic operations) and empty fields are not stored in the array. Also, if two or more columns of the result have the same column names, the later column will take precedence. In this situation, either call sesam_fetch_row() or make an alias for the column.

```
SELECT TBL1.COL AS FOO, TBL2.COL AS BAR FROM TBL1, TBL2
```

A special handling allows fetching "multiple field" columns (which would otherwise all have the same column names). For each column of a "multiple field", the index name is constructed by appending the string "(n)" where n is the sub-index of the multiple field column, ranging from 1 to its declared repetition factor. The indices are NOT zero based, in order to match the nomenclature used in the respective query syntax. For a column declared as:

```
CREATE TABLE ... ( ... MULTI(3) INT )
```

the associative indices used for the individual "multiple field" columns would be `"multi(1)"`, `"multi(2)"`, and `"multi(3)"` respectively.

Subsequent calls to **sesam_fetch_array()** would return the next (or prior, or n’th next/prior, depending on the scroll attributes) row in the result set, or `FALSE` if there are no more rows.

Esempio 1. SESAM fetch array

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table:
print "<TABLE BORDER>\n";
while (($row = sesam_fetch_array ($result)) && count ($row) > 0) {
    print " <TR>\n";
    print " <TD>".htmlspecialchars ($row["firstname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["lastname"])."</TD>\n";
    print " <TD>".htmlspecialchars ($row["phoneno"])."</TD>\n";
    print " </TR>\n";
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: sesam_fetch_row() which returns an indexed array.

sesam_fetch_result (PHP 3 CVS only)

Return all or part of a query result

mixed **sesam_fetch_result** (string result_id [, int max_rows]) \linebreak

Returns a mixed array with the query result entries, optionally limited to a maximum of *`max_rows`* rows. Note that both row and column indexes are zero-based.

Tabella 1. Mixed result set returned by sesam_fetch_result()

Array Element	Contents
int $arr["count"]	number of columns in result set (or zero if this was an "immediate" query)
int $arr["rows"]	number of rows in result set (between zero and *`max_rows`*)

Array Element	Contents
bool $arr["truncated"]	TRUE if the number of rows was at least *max_rows*, FALSE otherwise. Note that even when this is TRUE, the next **sesam_fetch_result()** call may return zero rows because there are no more result entries.
mixed $arr[col][row]	result data for all the fields at row(row) and column(col), (where the integer index row is between 0 and $arr["rows"]-1, and col is between 0 and $arr["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php isset() function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Note that the amount of memory used up by a large query may be gigantic. Use the *max_rows* parameter to limit the maximum number of rows returned, unless you are absolutely sure that your result will not use up all available memory.

See also: sesam_fetch_row(), and sesam_field_array() to check for "multiple fields". See the description of the sesam_query() function for a complete example using **sesam_fetch_result()**.

sesam_fetch_row (PHP 3 CVS only)

Fetch one row as an array

array **sesam_fetch_row** (string result_id [, int whence [, int offset]]) \linebreak

Returns an array that corresponds to the fetched row, or FALSE if there are no more rows.

The number of columns in the result set is returned in an associative array element $array["count"]. Because some of the result columns may be empty, the count() function can not be used on the result row returned by **sesam_fetch_row()**.

result_id is a valid result id returned by sesam_query() (select type queries only!).

whence is an optional parameter for a fetch operation on "scrollable" cursors, which can be set to the following predefined constants:

Tabella 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially (after fetch, the internal default is set to SESAM_SEEK_NEXT)
1	SESAM_SEEK_PRIOR	read sequentially backwards (after fetch, the internal default is set to SESAM_SEEK_PRIOR)

Value	Constant	Meaning
2	SESAM_SEEK_FIRST	rewind to first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	seek to last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	seek to absolute row number given as *offset* (Zero-based. After fetch, the internal default is set to SESAM_SEEK_ABSOLUTE, and the internal offset value is auto-incremented)
5	SESAM_SEEK_RELATIVE	seek relative to current scroll position, where *offset* can be a positive or negative offset value.

This parameter is only valid for "scrollable" cursors.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. If the *whence* parameter is omitted, the global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT, and settable by sesam_seek_row()) are used. If *whence* is supplied, its value replaces the global default.

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE. This parameter is only valid for "scrollable" cursors.

sesam_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array (indexed by values between 0 and $array["count"]-1). Fields may be empty, so you must check for the existence of a field by using the php isset() function. The type of the returned fields depend on the respective SQL type declared for its column (see SESAM overview for the conversions applied). SESAM "multiple fields" are "inlined" and treated like a sequence of columns.

Subsequent calls to **sesam_fetch_row()** would return the next (or prior, or n'th next/prior, depending on the scroll attributes) row in the result set, or FALSE if there are no more rows.

Esempio 1. SESAM fetch rows

```
<?php
$result = sesam_query ("SELECT * FROM phone\n".
                       "  WHERE LASTNAME='".strtoupper($name)."'\n".
                       "  ORDER BY FIRSTNAME", 1);
if (!$result) {
    ... error ...
}
// print the table in backward order
print "<TABLE BORDER>\n";
$row = sesam_fetch_row ($result, SESAM_SEEK_LAST);
while (is_array ($row)) {
    print " <TR>\n";
    for ($col = 0; $col < $row["count"]; ++$col) {
        print "  <TD>".htmlspecialchars ($row[$col])."</TD>\n";
```

```
    }
    print " </TR>\n";
    // use implied SESAM_SEEK_PRIOR
    $row = sesam_fetch_row ($result);
}
print "</TABLE>\n";
sesam_free_result ($result);
?>
```

See also: sesam_fetch_array() which returns an associative array, and sesam_fetch_result() which returns many rows per invocation.

sesam_field_array (PHP 3 CVS only)

Return meta information about individual columns in a result

array **sesam_field_array** (string result_id) \linebreak

result_id is a valid result id returned by sesam_query().

Returns a mixed associative/indexed array with meta information (column name, type, precision, ...) about individual columns of the result after the query associated with *result_id*.

Tabella 1. Mixed result set returned by sesam_field_array()

Array Element	Contents
int $arr["count"]	Total number of columns in result set (or zero if this was an "immediate" query). SESAM "multiple fields" are "inlined" and treated like the respective number of columns.
string $arr[col]["name"]	column name for column(`col`), where `col` is between 0 and `$arr["count"]-1`. The returned value can be the empty string (for dynamically computed columns). SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same column name.
string $arr[col]["count"]	The "count" attribute describes the repetition factor when the column has been declared as a "multiple field". Usually, the "count" attribute is 1. The first column of a "multiple field" column however contains the number of repetitions (the second and following column of the "multiple field" contain a "count" attribute of 1). This can be used to detect "multiple fields" in the result set. See the example shown in the sesam_query() description for a sample use of the "count" attribute.

Array Element	Contents
string $arr[col]["type"]	php variable type of the data for column(`col`), where `col` is between 0 and `$arr["count"]-1`. The returned value can be one of • integer • float • string depending on the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same php type.
string $arr[col]["sqltype"]	SQL variable type of the column data for column(`col`), where `col` is between 0 and `$arr["count"]-1`. The returned value can be one of • "CHARACTER" • "VARCHAR" • "NUMERIC" • "DECIMAL" • "INTEGER" • "SMALLINT" • "FLOAT" • "REAL" • "DOUBLE" • "DATE" • "TIME" • "TIMESTAMP" describing the SQL type of the result. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same SQL type.
string $arr[col]["length"]	The SQL "length" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "length" attribute is used with "CHARACTER" and "VARCHAR" SQL types to specify the (maximum) length of the string variable. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same length attribute.

Array Element	Contents
string $arr[col]["precision"]	The "precision" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "precision" attribute is used with numeric and time data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same precision attribute.
string $arr[col]["scale"]	The "scale" attribute of the SQL variable in column(`col`), where `col` is between 0 and `$arr["count"]-1`. The "scale" attribute is used with numeric data types. SESAM "multiple fields" are "inlined" and treated like the respective number of columns, each with the same scale attribute.

See the sesam_query() function for an example of the **sesam_field_array()** use.

sesam_field_name (PHP 3 CVS only)

Return one column name of the result set

int **sesam_field_name** (string result_id, int index) \linebreak

Returns the name of a field (i.e., the column name) in the result set, or `FALSE` on error.

For "immediate" queries, or for dynamic columns, an empty string is returned.

> **Nota:** The column index is zero-based, not one-based as in SESAM.

See also: sesam_field_array(). It provides an easier interface to access the column names and types, and allows for detection of "multiple fields".

sesam_free_result (PHP 3 CVS only)

Releases resources for the query

int **sesam_free_result** (string result_id) \linebreak

Releases resources for the query associated with *`result_id`*. Returns `FALSE` on error.

sesam_num_fields (PHP 3 CVS only)

Return the number of fields/columns in a result set

int **sesam_num_fields** (string result_id) \linebreak

After calling sesam_query() with a "select type" query, this function gives you the number of columns in the result. Returns an integer describing the total number of columns (aka. fields) in the current *result_id* result set or FALSE on error.

For "immediate" statements, the value zero is returned. The SESAM "multiple field" columns count as their respective dimension, i.e., a three-column "multiple field" counts as three columns.

See also: sesam_query() and sesam_field_array() for a way to distinguish between "multiple field" columns and regular columns.

sesam_query (PHP 3 CVS only)

Perform a SESAM SQL query and prepare the result

string **sesam_query** (string query [, bool scrollable]) \linebreak

Returns: A SESAM "result identifier" on success, or FALSE on error.

A "result_id" resource is used by other functions to retrieve the query results.

sesam_query() sends a query to the currently active database on the server. It can execute both "immediate" SQL statements and "select type" queries. If an "immediate" statement is executed, then no cursor is allocated, and any subsequent sesam_fetch_row() or sesam_fetch_result() call will return an empty result (zero columns, indicating end-of-result). For "select type" statements, a result descriptor and a (scrollable or sequential, depending on the optional boolean *scrollable* parameter) cursor will be allocated. If *scrollable* is omitted, the cursor will be sequential.

When using "scrollable" cursors, the cursor can be freely positioned on the result set. For each "scrollable" query, there are global default values for the scrolling type (initialized to: SESAM_SEEK_NEXT) and the scrolling offset which can either be set once by sesam_seek_row() or each time when fetching a row using sesam_fetch_row().

For "immediate" statements, the number of affected rows is saved for retrieval by the sesam_affected_rows() function.

See also: sesam_fetch_row() and sesam_fetch_result().

Esempio 1. Show all rows of the "phone" table as a html table

```
<?php
if (!sesam_connect ("phonedb", "demo", "otto"))
    die ("cannot connect");
$result = sesam_query ("select * from phone");
if (!$result) {
    $err = sesam_diagnostic();
    die ($err["errmsg"]);
}
echo "<TABLE BORDER>\n";
// Add title header with column names above the result:
if ($cols = sesam_field_array ($result)) {
    echo " <TR><TH COLSPAN=".$cols["count"].">Result:</TH></TR>\n";
    echo " <TR>\n";
    for ($col = 0; $col < $cols["count"]; ++$col) {
        $colattr = $cols[$col];
```

```
        /* Span the table head over SESAM's "Multiple Fields": */
        if ($colattr["count"] > 1) {
            echo "  <TH COLSPAN=".$colattr["count"].">".$colattr["name"].
                "(1..".$colattr["count"].")</TH>\n";
            $col += $colattr["count"] - 1;
        } else
            echo "  <TH>" . $colattr["name"] . "</TH>\n";
    }
    echo " </TR>\n";
}

do {
    // Fetch the result in chunks of 100 rows max.
    $ok = sesam_fetch_result ($result, 100);
    for ($row=0; $row < $ok["rows"]; ++$row) {
        echo " <TR>\n";
        for ($col = 0; $col < $ok["cols"]; ++$col) {
            if (isset($ok[$col][$row]))
                echo "  <TD>" . $ok[$col][$row] . "</TD>\n";
            } else {
                echo "  <TD>-empty-</TD>\n";
            }
        }
        echo " </TR>\n";
    }
}
while ($ok["truncated"]) { // while there may be more data
    echo "</TABLE>\n";
}
// free result id
sesam_free_result($result);
?>
```

sesam_rollback (PHP 3 CVS only)

Discard any pending updates to the SESAM database

bool **sesam_rollback** (void) \linebreak

Returns: TRUE on success, FALSE on errors

sesam_rollback() discards any pending updates to the database. Also affected are result cursors and result descriptors.

At the end of each script, and as part of the sesam_disconnect() function, an implied **sesam_rollback()** is executed, discarding any pending changes to the database.

See also: sesam_commit().

Esempio 1. Discarding an update to the SESAM database

```
<?php
if (sesam_connect ("mycatalog", "myschema", "otto")) {
    if (sesam_execimm ("INSERT INTO mytable VALUES (*, 'Small Test', <0, 8, 15>)")
        && sesam_execimm ("INSERT INTO othertable VALUES (*, 'Another Test', 1)"))
        sesam_commit();
    else
        sesam_rollback();
}
?>
```

sesam_seek_row (PHP 3 CVS only)

Set scrollable cursor mode for subsequent fetches

bool **sesam_seek_row** (string result_id, int whence [, int offset]) \linebreak

result_id is a valid result id (select type queries only, and only if a "scrollable" cursor was requested when calling sesam_query()).

whence sets the global default value for the scrolling type, it specifies the scroll type to use in subsequent fetch operations on "scrollable" cursors, which can be set to the following predefined constants:

Tabella 1. Valid values for *"whence"* parameter

Value	Constant	Meaning
0	SESAM_SEEK_NEXT	read sequentially
1	SESAM_SEEK_PRIOR	read sequentially backwards
2	SESAM_SEEK_FIRST	fetch first row (after fetch, the default is set to SESAM_SEEK_NEXT)
3	SESAM_SEEK_LAST	fetch last row (after fetch, the default is set to SESAM_SEEK_PRIOR)
4	SESAM_SEEK_ABSOLUTE	fetch absolute row number given as *offset* (Zero-based. After fetch, the default is set to SESAM_SEEK_ABSOLUTE, and the offset value is auto-incremented)

Value	Constant	Meaning
5	SESAM_SEEK_RELATIVE	fetch relative to current scroll position, where *offset* can be a positive or negative offset value (this also sets the default "offset" value for subsequent fetches).

offset is an optional parameter which is only evaluated (and required) if *whence* is either SESAM_SEEK_RELATIVE or SESAM_SEEK_ABSOLUTE.

sesam_settransaction (PHP 3 CVS only)

Set SESAM transaction parameters

bool **sesam_settransaction** (int isolation_level, int read_only) \linebreak

Returns: TRUE if the values are valid, and the **settransaction()** operation was successful, FALSE otherwise.

sesam_settransaction() overrides the default values for the "isolation level" and "read-only" transaction parameters (which are set in the SESAM configuration file), in order to optimize subsequent queries and guarantee database consistency. The overridden values are used for the next transaction only.

sesam_settransaction() can only be called before starting a transaction, not after the transaction has been started already.

To simplify the use in php scripts, the following constants have been predefined in php (see SESAM handbook for detailed explanation of the semantics):

Tabella 1. Valid values for *"Isolation_Level"* parameter

Value	Constant	Meaning
1	SESAM_TXISOL_READ_UNCOMMITTED	Read Uncommitted
2	SESAM_TXISOL_READ_COMMITTED	Read Committed
3	SESAM_TXISOL_REPEATABLE_READ	Repeatable Read
4	SESAM_TXISOL_SERIALIZABLE	Serializable

Tabella 2. Valid values for *"Read_Only"* parameter

Value	Constant	Meaning
0	SESAM_TXREAD_READWRITE	Read/Write
1	SESAM_TXREAD_READONLY	Read-Only

The values set by **sesam_settransaction()** will override the default setting specified in the SESAM configuration file.

Esempio 1. Setting SESAM transaction parameters

```
<?php
sesam_settransaction (SESAM_TXISOL_REPEATABLE_READ,
                      SESAM_TXREAD_READONLY);
?>
```

XCI. Funzioni di gestione della sessione

Il supporto delle sessioni in PHP consiste nel mantenere certi dati attraverso accessi successivi.Questo vi dà la capacità di costruire applicazioni più consone alle vostre esigenze e di accrescere le qualità del vostro sito web.

Se avete dimestichezza con la gestione delle sessioni di PHPLIB, noterete che alcuni concetti sono simili al supporto dele sessioni in PHP.

Al visitatore che accede al vostro sito web viene assegnato un id unico, il cosidetto id di sessione. Questo viene registrato in un cookie sul lato utente o è propagato tramite l'URL.

Il supporto delle sessioni vi permette di registrare numeri arbitrari di variabili che vengono preservate secondo richiesta.Quando un visitatore accede al vostro sito, PHP controllerà automaticamente (se session.auto_start è settato a 1) o su vostra richiesta (esplicitamente tramite session_start() o implicitamente tramite session_register()) se uno specifico id di sessione sia stato inviato con la richiesta. In questo caso , il precedente ambiente salvato viene ricreato.

Tutte le variabili registrate vengono serializzate dopo che la richiesta è finita.Le variabili registrate che non sono definite vengono marcate come indefinite. All'accesso successivo, queste non vengono definite dal modulo di sessione fino a quando l'utente non le definisce più tardi.

La configurazione di `track_vars` e `register_globals` influenza come le variabili di sessione vengono memorizzate una e più volte.

> **Nota:** In PHP 4.0.3, `track_vars` è sempre attiva.

> **Nota:** In PHP 4.1.0, `$_SESSION` è disponibile come variabile globale proprio come `$_POST`, `$_GET`, `$_REQUEST` e così via. `$_SESSION` non è sempre globale come `$HTTP_SESSION_VARS`. Per questo motivo, il termine `global` non dovrebbe essere usato per `$_SESSION`.

Se `track_vars` è attiva e `register_globals` non è attiva, solo i membri dell'array associativo globale `$HTTP_SESSION_VARS` possono essere registrati come variabili di sessione. Le variabili di sessione ripristinate saranno disponibili nell'array `$HTTP_SESSION_VARS`.

Esempio 1. Registrare una variabile con `track_vars` attiva

```
<?php
if (isset($HTTP_SESSION_VARS['count'])) {
   $HTTP_SESSION_VARS['count']++;
}
else {
   $HTTP_SESSION_VARS['count'] = 0;
}
?>
```

L'uso di `$_SESSION` (o `$HTTP_SESSION_VARS` con PHP 4.0.6 o precedente) è raccomandato per sicurezza e leegibilità del codice.Con `$_SESSION` o `$HTTP_SESSION_VARS`, non c'è bisogno di

usare le funzioni session_register()/session_unregister()/session_is_registered(). Gli utenti possono accedere alla variabile di sessione come a una variabile normale.

Esempio 2. Registrare una variabile con $_SESSION.

```
<?php
// Use $HTTP_SESSION_VARS with PHP 4.0.6 or less
if (!isset($_SESSION['count'])) {
    $_SESSION['count'] = 0;
} else {
    $_SESSION['count']++;
}
?>
```

Esempio 3. Resettare una variabile con $_SESSION.

```
<?php
// Use $HTTP_SESSION_VARS with PHP 4.0.6 or less
unset($_SESSION['count']);

?>
```

Se `register_globals` è attiva, allora tutte le variabili globali possono essere registrate come variabili di sessione e le variabili di sessione saranno ripristinate in corrispondenza delle variabili globali. Dal momento che PHP ha bisogno di sapere quali variabili globali sono registrate come variabili di sessione , gli utenti devono registrare le variabili con la funzione session_register() mentre `$HTTP_SESSION_VARS`/`$_SESSION` non ha bisogno di usare session_register().

Cautela

Se state usando `$HTTP_SESSION_VARS`/`$_SESSION` e `register_globals` non è attiva, non usate session_register(), session_is_registered() e session_unregister().

Se attivate `register_globals`, session_unregister() dovrebbe essere usata dal momento in cui le variabili di sessione vengono registrate come variabili globali quando i dati di sessione vengono deserializzati. Disattivare `register_globals` è raccomandato sia per motivi di sicurezza che di prestazione.

Esempio 4. Registrare una variabile con `register_globals` attiva

```
<?php
if (!session_is_registered('count')) {
    session_register("count");
    $count = 0;
}
```

```
else {
    $count++;
}
?>
```

Se entrambe `track_vars` e `register_globals` sono attivate, allora le variabili globali e le entrate di `$HTTP_SESSION_VARS`/`$_SESSION` riporteranno lo stesso valore per variabili già registrate.

Se l'utente usa session_register() pre registrare una variabile di sessione, `$HTTP_SESSION_VARS`/`$_SESSION` non avranno questa variabile nell'array fino a che non sarà caricata dall'archivio di sessione.(i.e. fino alla prossima richiesta)

Ci sono due metodi per propagare l'id di sessione:

- I Cookies
- Un parametro dell'URL

Il modulo di sessione supporta entrambi i metodi. I cookies sono ottimi, ma dal momento che possono non essere a disposizione (i clients non sono costretti ad accettarli), non possiamo dipendere da questi. Il secondo metodo incorpora l'id di sessione direttamente negli URL.

PHP ha la capacità di farlo in modo trasparente quando compilato con `--enable-trans-sid`. Se attivate questa opzione, gli URL relativi saranno modificati per contenere l'id di sessione automaticamente. In alternativa, potete usare la costante Alternatively, you can use the constant `SID` che è definita, se il client non ha mandato il cookie appropriato. `SID` può avere la forma di `session_name=session_id` o può essere una stringa vuota.

L'esempio seguente dimostra come registrare una variabile e come collegare una pagina all'altra correttamente usando SID.

Esempio 5. Contare il numero di accessi di un singolo utente

```
<?php
if (!session_is_registered('count')) {
    session_register('count');
    $count = 1;
}
else {
    $count++;
}
?>

Salve visitatore , hai visitato questa pagina <?php echo $count; ?> times.<p>;

<?php
# il <?php echo SID?> (<?=SID?> può essere usato se short tag è attivo)
# è necessario per preservare l'id di sessione
# nel caso incui l'utente abbia disattivato i cookies
?>

Per continuare, <A HREF="nextpage.php?<?php echo SID?>">clicca qui</A>
```

Il `<?=SID?>` non è necessario, se `--enable-trans-sid` è stato usato per compilare PHP.

> **Nota:** Gli URL non relativi si presume che puntino a siti esterni e quindi non hanno il SID , perchè sarebbe rischioso per la sicurezza propagare il SID a un altro server.

Per implementare l'archiviazione in database , o qualsiasi altro metodo di archiviazione, avete bisogno di usare session_set_save_handler() per creare un set di funzioni di archiviazione a livello utente.

Il sistema di gestione delle sessioni supporta un numero di opzioni di configurazione che potete posizionare nel vostro file php.ini. Ne daremo una breve spiegazione.

- `session.save_handler` definisce il nome dell'handler che è usato per archiviare e rilasciare i dati associati a una sessione. Di default è `files`.
- `session.save_path` definisce l'argomento che è passato all'handler di sessione. Se scegliete handler files di default , questo è il percorso dove i files vengono creati. Di default è `/tmp`. Se la profondità del percorso `session.save_path` è più di 2, l'accumulo (gc) non sarà effettuato.

> **Attenzione**
>
> Se lasciate questo settato a directory leggibile da tutti , come If you leave this set to a world-readable directory, such as `/tmp` (il default), altri utenti sul potrebbero essere in grado di dirottare le sessioni prendendo la lista dei files in quella directory.

- `session.name` specifica il nome della sessione che è usata come nome del cookie. Dovrebbe contenere solo caratteri alfanumerici. Di default è `PHPSESSID`.
- `session.auto_start` specifica se il modulo di sessione inizia una sessione automaticamente su richiesta iniziale. Di default è `0` (disattivata).
- `session.cookie_lifetime` specifica il tempo di vita insecondi del cookie che viene mandato al browser. Il valore 0 significa "fino a che il browser viene chiuso". Di default è `0`.
- `session.serialize_handler` definisce il nome dell'handler che è usato per serializzare/deserializzare i dati. Al momento, un formato interno di PHP(nome `php`) e WDDX è supportato (nome `wddx`). WDDX è solo disponibile, se PHP è compilato con WDDX support. Il defailt è `php`.
- `session.gc_probability` specifica la probabilità , in percentuale ,che la routine gc (garbage collection) sia cominciata ad ogni richiesta in percentuale. Di default è `1`.
- `session.gc_maxlifetime` specifica il numero di secondi dopo i quali i dati saranno considerati 'spazzatura' e cancellati.
- `session.referer_check` contiene la sottostringa con cui volete controllare ogni HTTP referer. Se il referer è stato mandato dal client e la sottostringa non è stata trovata, l'id incorporato nella

sessione verrà marcato come non valido. Il default è una stringa vuota.

- `session.entropy_file` dà un percorso a una risorsa esterna (file) che sarà usata come una addizionale sorgente entropica nella crazione dell'id di sessione. Esempi sono `/dev/random` o `/dev/urandom` che sono disponibili sulla maggior parte dei sistemi Unix.
- `session.entropy_length` specifica il numero di bytes che saranno letti dal file specificato sopra. Di default è `0` (disattivato).
- `session.use_cookies` specifica se il modulo userà i cookies per archiviare l'id di sessione sul lato client. Di default è `1` (attivo).
- `session.cookie_path` specifica il percorso da stabilire in session_cookie. Di default è `/`.
- `session.cookie_domain` specifica il dominio settato in session_cookie. Di default è niente.
- `session.cache_limiter` specifica il metodo di controllo della cache da usare per le pagine di sessione (none/nocache/private/private_no_expire/public). Di default è `nocache`.
- `session.cache_expire` specifica il tempo-di-vita , in minuti , delle pagine nella cache, questo non ha effetto sul limitatore nocache. Di default è `180`.
- `session.use_trans_sid` specifica se il supporto sid trasparente è attivato o no se attivato compilandolo con `--enable-trans-sid`. Di default è `1` (attivo).
- `url_rewriter.tags` specifica quali html tags sono riscritti per includere l'id di sessione se il supporto sid trasparente è attivato. Di default è `a=href,area=href,frame=src,input=src,form=fakeentry`

> **Nota:** L'handling di sessione è stato aggiunto in PHP 4.0.

session_cache_expire (PHP 4 >= 4.2.0)

Restituisce l'espirazione della cache corrente

int **session_cache_expire** ([int new_cache_expire]) \linebreak

session_cache_expire() restituisce l'espirazione della cache corrente. Se è data *`new_cache_expire`* , l'espirazione della cache corrente è rimpiazzata da *`new_cache_expire`*.

session_cache_limiter (PHP 4 >= 4.0.3)

Assume o imposta il limitatore di cache corrente

string **session_cache_limiter** ([string cache_limiter]) \linebreak

session_cache_limiter() restituisce il nome del limitatore di cache corrente. Se *`cache_limiter`* è specificato, il nome del limitatore di cache corrente viene cambiato nel nuovo valore.

Il limitatore di cache controlla la cache degli headers HTTP mandati al client. Questi headers determinano i modi in cui il contenuto della pagina possono essere depositati. Impostando il limitatore di cache a `nocache`, per esempio, non permetterebbe nessun caching lato client. Un valore di `public`, invece, permetterebbe il caching. Può anche essere impostato a `private`, che è leggermente più restrittivo di `public`.

Nella modalità `private` , l'header Expire mandato al client, potrebbe causare confusione per alcuni browser incluso Mozilla. Potete evitare questo problema con la modalità `private_no_expire` . In questo modo l'header Expire non viene mai spedito al client.

Nota: `private_no_expire` è stato aggiunto in PHP 4.2.0dev.

Il limitatore di cache è resettato al valore di default archiviato in `session.cache_limiter` alla richiesta iniziale. Per questo motivo, avete bisogno di chiamare **session_cache_limiter()** per ogni richiesta (e prima che session_start() sia chiamata).

Esempio 1. session_cache_limiter() esempi

```
<?php

# set the cache limiter to 'private'

session_cache_limiter('private');
$cache_limiter = session_cache_limiter();

echo "Il limitatore di cache è adesso impostato a $cache_limiter<p>";
?>
```

session_decode (PHP 4 >= 4.0.0)

Decodifica i dati di sessione da una stringa

bool **session_decode** (string data) \linebreak

session_decode() decodifica i dati di sessione in `data`, impostando le varibili archiviate nella sessione.

session_destroy (PHP 4 >= 4.0.0)

Distrugge tutti i dati registrati in una sessione

bool **session_destroy** (void) \linebreak

session_destroy() distrugge tutti i dati associati alla sessione corrente. Non desetta nessuna delle variabili globali associate alla sessione o desetta il cookie di sessione.

Questa funzione ritorna `TRUE` in caso di successo e `FALSE` in caso di fallimento nel distruggere i dati di sessione.

Esempio 1. Distruggere una sessione

```
<?php

// Inizializza la sessione.
// Se state usando session_name("qualcosa"), non dimenticatevelo adesso!
session_start();
// Desetta tutte le variabili di sessione.
session_unset();
// Infine , distrugge la sessione.
session_destroy();

?>
```

Esempio 2. Distruggere una sessione con $_SESSION

```
<?php

// Inizializza la sessione.
// Se state usando session_name("qualcosa"), non dimenticatevelo adesso!
session_start();
// Desetta tutte le variabili di sessione.
$_SESSION = array();
// Infine distrugge la sessione.
session_destroy();
```

```
?>
```

session_encode (PHP 4 >= 4.0.0)

Codifica i dati della sessione corrente in una stringa

string **session_encode** (void) \linebreak

session_encode() restituisce una stringa con i contenuti della sessione corrente codificati.

session_get_cookie_params (PHP 4 >= 4.0.0)

Restituisce i parametri del cookie di sessione

array **session_get_cookie_params** (void) \linebreak

La funzione **session_get_cookie_params()** restituisce un con le informazioni sul cookie di sessione corrente, l'array contiene i seguenti elementi:

- "lifetime" - La durata del cookie.
- "path" - Il percorso dove l'informazione è archiviata.
- "domain" - Il dominio di validità del cookie.
- "secure" - Il cookie dovrebbe essere spedito solo attraverso connessioni sicure. (Questo elemento è stato aggiunto in PHP 4.0.4.)

session_id (PHP 4 >= 4.0.0)

Assume o imposta l'id di sessione corrente

string **session_id** ([string id]) \linebreak

session_id() restituisce l'id di sessione per la sessione corrente. Se `id` è specificato, sostituirà l'id di sessione corrente.

La costante SID può essere usata anche per fornire nome e id correnti di sessione come una stringa fatta in modo che si possa aggiungere agli Url.

session_is_registered (PHP 4 >= 4.0.0)

Scopre se una variabile è registrata nella sessione

bool **session_is_registered** (string name) \linebreak

session_is_registered() restituisce `TRUE` se c'è una variabile con il nome *`name`* registrato nella sessione corrente.

> **Nota:** Se è usata `$_SESSION` (o `$HTTP_SESSION_VARS` per PHP 4.0.6 o inferiore), usate isset() per controllare che una variabile sia registrata in `$_SESSION`.

> **Cautela**
>
> Se state usando `$HTTP_SESSION_VARS`/`$_SESSION`, non usate session_register(), **session_is_registered()** e session_unregister().

session_module_name (PHP 4 >= 4.0.0)

Assume o imposta il corrente modulo di sessione

string **session_module_name** ([string module]) \linebreak

session_module_name() restituisce il nome del corrente modulo di sessione. Se *`module`* è specificato, sarà invece usato quel modulo.

session_name (PHP 4 >= 4.0.0)

Dà e/o stabilisce il nome della sessione corrente

string **session_name** ([string name]) \linebreak

session_name() ritorna il nome della sessione corrente. Se *`name`* è specificato, il nome della sessione corrente viene cambiato al suo valore.

Il nome della sessione riporta l'id nei coookies e negli URl. Dovrebbe contenere solo caratteri alfanumerici; dovrebbe essere corto e descrittivo (i.e. per utenti con l'avviso di cookie attivo). Il nome di sessione è resettato al valore di default archiviato in `session.name` quando avviene la richiesta iniziale. Tuttavia, avete bisogno di chiamare **session_name()** per ogni richiesta (e prima vengono chiamate session_start() o session_register()).

Esempio 1. session_name() esempi

```
<?php

// imposta il nome di sessione a WebsiteID

$previous_name = session_name("WebsiteID");

echo "Il precedente nome di sessione è $previous_name<p>";
?>
```

session_readonly (unknown)

Begin session - reinitializes freezed variables, but no writeback on request end

void **session_readonly** (void) \linebreak

Read in session data without locking the session data. Changing session data is not possible, but frameset performance will be improved.

session_register (PHP 4 >= 4.0.0)

Registra una o più variabili con la sessione corrente

bool **session_register** (mixed name [, mixed ...]) \linebreak

session_register() accetta un numero di argomenti variabile, ognuno dei quali può sia essere una stringa contenente il nome di una variabile o un array che contiene i nomi delle variabili o altri arrays. Per ogni nome, **session_register()** registra la variabile globale con quel nome nella sessione corrente.

> **Cautela**
>
> Questo registra un variabile *global*. Se volete registrare una variabile di sessione interna a una funzione, avete bisogno di assicurarvi di farla globale usando **global()** o usate gli arrays di sessione come scritto sotto.

> **Cautela**
>
> Se state usando `$HTTP_SESSION_VARS`/`$_SESSION`, non usate **session_register()**, session_is_registered() e session_unregister().

Questa funzione restituisce `TRUE` quando tutte le variabili sono registrate con successo nella sessione.

Se session_start() non è stata chiamata prima che questa funzione venga chiamata, avverrà una chiamata imlicita senza parametri a session_start().

Potete anche creare una variabile di sessione semplicemente impostando l'appropriato membro di `$HTTP_SESSION_VARS` o `$_SESSION` (PHP >= 4.1.0) array.

```
$barney = "Una grande torta fiammeggiante.";
session_register("barney");

$HTTP_SESSION_VARS["zim"] = "Mars attack.";

# the auto-global $_SESSION array was introduced in PHP 4.1.0
$_SESSION["spongebob"] = "Ha i pantaloni a quadri.";
```

> **Nota:** Non è possibile registrare risorse variabili in una sessione. Per esempio, non potete creare una connessione a un database e archiviare l'id della connessione come una variabile di sessione e aspettarvi che la connessione sia ancora valida la prossima volta che la sessione viene riastabilita. Le funzioni PHP che restituiscono una risorsa sono identificate avendo un tipo di restituzione `resource` nelle loro definizioni di funzione. Una lista di funzioni che restituisce risorse è disponibile nell'appendice resource types.
>
> Se viene usata `$_SESSION` (o `$HTTP_SESSION_VARS` per PHP 4.0.6 or inferiore), assegna la variabile a `$_SESSION`. i.e. $_SESSION['var'] = 'ABC';

Vedere anche session_is_registered() e session_unregister().

session_save_path (PHP 4 >= 4.0.0)

Assume o stabilisce il percorso di salvataggio sessione corrente

string **session_save_path** ([string path]) \linebreak

session_save_path() restituisce il percorso della directory corrente usata per salvare i dati di sessione. Se *path* è specificato, il percorso in quale i dati vengono salvati verrà cambiata.

> **Nota:** Su alcuni sistemi operativi, potreste voler specificare un percorso su un filesystem che gestisce molti piccoli files in modo efficiente. Per esempio, su Linux, reiserfs potrebbe garantire una migliore prestazione di ext2fs.

session_set_cookie_params (PHP 4 >= 4.0.0)

Imposta i parametri del cookie di sessione

void **session_set_cookie_params** (int lifetime [, string path [, string domain]]) \linebreak

Imposta i parametri del cookie definiti nel file php.ini. L'effetto di questa funzione dura solo per la durata dello script.

session_set_save_handler (PHP 4 >= 4.0.0)

Imposta le funzioni di archiviazione sessioni a livello utente

void **session_set_save_handler** (string open, string close, string read, string write, string destroy, string gc) \linebreak

session_set_save_handler() imposta le funzioni di archiviazione sessioni che sono usate per archiviare e riutilizzare i dati associati a una sessione. Ciò non è molto utile quando un altro metodo di archiviazione è preferito a quelli forniti dalle sessioni PHP. i.e. L'archiviazione dei dati di sessione in un database locale.

> **Nota:** Dovete impostare l'opzione di configurazione `session.save_handler` per `user` nel vostro file php.ini perchè **session_set_save_handler()** abbia effetto.

> **Nota:** L'handler "write" non viene eseguito fino a che l'output stream non viene chiuso. In questo modo, l'output di espressioni di debugging nell'hanlder "write" non si vedrà mai nel browser. Se l'output di debugging è necessario, è consigliabile che l'output del debug venga scritto in un file.

Il seguente esempio fornisce l'archiviazione di sessione basata su file simile al solito gestore di salvataggio di sessioni PHP `files`. Questo esempio potrebbe essere facilmente esteso per coprire l'archiviazione in database usando il vostro sistema database favorito con supporto PHP.

La funzione di lettura deve restituire sempre un valore stringa perchè il save handler funzioni a dovere. Restituisce una stringa vuota se non ci sono dati da leggere. I valori restituiti da altri handlers sono convertiti in espressioni booleane. TRUE per successo, FALSE in caso di fallimento.

Esempio 1. session_set_save_handler() esempio

```
<?php
function open ($save_path, $session_name) {
  global $sess_save_path, $sess_session_name;

  $sess_save_path = $save_path;
  $sess_session_name = $session_name;
  return(true);
}

function close() {
  return(true);
}

function read ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "r")) {
    $sess_data = fread($fp, filesize($sess_file));
    return($sess_data);
  } else {
    return(""); // Deve restituire "" qui.
  }

}

function write ($id, $sess_data) {
  global $sess_save_path, $sess_session_name;
```

```
  $sess_file = "$sess_save_path/sess_$id";
  if ($fp = @fopen($sess_file, "w")) {
    return(fwrite($fp, $sess_data));
  } else {
    return(false);
  }

}

function destroy ($id) {
  global $sess_save_path, $sess_session_name;

  $sess_file = "$sess_save_path/sess_$id";
  return(@unlink($sess_file));
}

/**********************************************
 * ATTENZIONE - Qui avete bisogno di implementare qualche *
 * sorta di routine per il cestinaggio.  *
 *********************************************/
function gc ($maxlifetime) {
  return true;
}

session_set_save_handler ("open", "close", "read", "write", "destroy", "gc");

session_start();

// proceed to use sessions normally

?>
```

session_start (PHP 4 >= 4.0.0)

Inizializza i dati di sessione

bool **session_start** (void) \linebreak

session_start() creauna sessione (o riprende quella corrente basata sull'id di sessione che viene passato attraverso una variabile GET o un cookie.

Se volete usare usare una sessione con un nome, dovete chiamare session_name() prima di **session_start()**.

Questa funzione ritorna sempre `TRUE`.

> **Nota:** Se state usando una sessione basata sui cookie, dovete chiamare **session_start()** prima di qualsiasi altro output al browser.

session_start() registrerà un handler interno di output per riscrivere l'URL quando `trans-sid` è attivato. Se l'utente usa `ob_gzhandler` o come con ob_start(), l'ordine dell'handler di output è importante per un giusto output. Per esempio, l'utente deve registrare `ob_gzhandler` prima che la sessione cominci.

> **Nota:** L'uso di `zlib.output_compression` è raccomandato più che di `ob_gzhandler`

session_unregister (PHP 4 >= 4.0.0)

Deregistra una variabile dalla sessione corrente

bool **session_unregister** (string name) \linebreak

session_unregister() deregistra (dimentica) la variabile globale con nome *name* dalla sessione corrente.

Questa funzione restituisce `TRUE` quando la variabile viene deregistrata con successo dalla sessione.

> **Nota:** Se viene usata `$_SESSION` (o `$HTTP_SESSION_VARS` per PHP 4.0.6 o inferiore), usate unset() per deregistrare una variabile di sessione.

> **Cautela**
>
> Questa funzione non deimposta la corrispondente variabile globale per *name*, impedisce solo che la variabile venga salvata come parte della sessione. Dovete chiamare unset() per rimuovere la variabile globale corrispondente.

> **Cautela**
>
> Se state usando `$HTTP_SESSION_VARS`/`$_SESSION`, non usate session_register(), session_is_registered() e **session_unregister()**.

session_unset (PHP 4 >= 4.0.0)

Libera tutte le variabili di sessione

void **session_unset** (void) \linebreak

La funzione **session_unset()** libera tutte le variabili di sessione correntemente registrate.

> **Nota:** Se è usata `$_SESSION` (o `$HTTP_SESSION_VARS` per PHP 4.0.6 o inferiore) è, usate unset() per deregistrare una variabile di sessione. i.e. $_SESSION = array();

session_write_close (PHP 4 >= 4.0.4)

Scrive i dati di sessione e termina la sessione

void **session_write_close** (void) \linebreak

Termina la sessione corrente e archivia i dati di sessione.

I dati di sessione sono di solito archiviati dopo che il vostro script è terminato senza il bisogno di chiamare **session_write_close()**, ma poichè i dati di sessione vengono bloccati per prevenire scritture contemporanee solo uno script può operare su una sessione in qualsiasi momento. Quando utilizzerete i framesets assieme alla sessione vedrete che i frames vengono caricati uno per uno a causa di questo bloccaggio. Potete ridurre il tempo necessario per caricare tutti i frames terminando la sessione appena tutti i cambi alle variabili di sessione sono stati fatti.

XCII. Funzioni relative alla memoria condivisa

Shmop è un set di funzioni di semplice utilizzo che permettono al PHP di leggere, scrivere, creare e cancellare i segmenti di memoria condivisa di Unix. Queste funzioni non sono attive sui sistemi Windows, dato che quest'ultimo non supporta la memoria condivisa. Per utilizzare shmop si deve compilare il PHP con il parametro --enable-shmop nella linea di configurazione.

> **Nota:** Nella versione 4.0.3 di PHP queste funzioni hanno il prefisso `shm` anzichè `shmop`.

Esempio 1. Descrizione delle operazioni con la memoria condivisa

```
<?php

// Crea un blocco di memoria condivisa di 100 byte con id 0xff3
$shm_id = shmop_open(0xff3, "c", 0644, 100);
if(!$shm_id) {
echo "Non si riesce a creare il segmento di memoria condivisa\n";
}

// Ottiene la dimensione del blocco di memoria
$shm_size = shmop_size($shm_id);
echo "Dimesione blocco creato: ".$shm_size. ".\n";

// Scrittura di una stringa di test nella memoria condivisa
$shm_bytes_written = shmop_write($shm_id, "my shared memory block", 0);
if($shm_bytes_written != strlen("my shared memory block")) {
echo "Non si riesce a scrivere tutti i dati\n";
}

// Ora si rilegge la stringa
$my_string = shmop_read($shm_id, 0, $shm_size);
if(!$my_string) {
echo "Non si riesce a leggere dalla memoria condivisa\n";
}
echo "I dati presenti nella memoria condivisa sono: ".$my_string."\n";

// Ora si cancella il blocco e si chiude il segmento di memoria condivisa
if(!shmop_delete($shm_id)) {
echo "Non si riesce a marcare il blocco di memoria condivisa per la cancellazione.";
}
shmop_close($shm_id);

?>
```

shmop_close (PHP 4 >= 4.0.4)

Chiusura di un blocco di memoria condivisa

int **shmop_close** (int shmid) \linebreak

Si utilizza la funzione **shmop_close()** per chiudere un segmento di memoria condivisa.

La funzione **shmop_close()** ha un solo parametro, shmid, che è l'identificativo del blocco di memoria condivisa creato da shmop_open().

Esempio 1. Chiusura di un blocco di memoria condivisa

```
<?php
shmop_close($shm_id);
?>
```

In questo esempio si chiude il blocco di memoria condivisa identificata da `$shm_id`.

shmop_delete (PHP 4 >= 4.0.4)

Cancella un blocco di memoria condivisa

int **shmop_delete** (int shmid) \linebreak

La funzione **shmop_delete()** viene utilizzata per cancellare un blocco di memoria condivisa.

La funzione **shmop_delete()** ha un solo parametro, shmid, che è l'identificativo del blocco di memoria condiviso creato da shmop_open(). Se la funzione ha successo restituisce 1, altrimenti 0.

Esempio 1. Cancellazione di un segmento di memoria condivisa

```
<?php
shmop_delete($shm_id);
?>
```

In questo esempio si cancella il segmento di memoria condivisa identificato da `$shm_id`.

shmop_open (PHP 4 >= 4.0.4)

Crea oppure apre un segmento di memoria condivisa

int **shmop_open** (int key, string flags, int mode, int size) \linebreak

La funzione **shmop_open()** può creare oppure aprire un segmento di memoria condivisa.

La funzione **shmop_open()** utilizza 4 parametri: key, indica l'identificativo di sistema per il segmento di memoria condivisa, questo parametro può essere passato come numero decimale o esadecimale. Il secondo parametro è un flag che può assumere i seguenti valori:

- "a" per accesso (SHM_RDONLY per shmat), usare questo flag quando occorre aprire un segmento di memoria condivisa esistente in sola lettura
- "c" per creazione (IPC_CREATE), usare questo flag quando si ha la necessità di creare un nuovo segmento di memoria condivisa oppure, se esiste già un segmento con la medesima chiave, tentare di aprirlo in lettura e scrittura
- "w" per accesso in lettura & scrittura, usare questo flag quando si deve accedere al segmento di memoria condivisa in lettura e scrittura, nella maggior parte dei casi si usa questo flag.
- "n" per creare un nuovo segmento (IPC_CREATE|IPC_EXCL), usare questo flag quando si vuole creare un nuovo segmento di memoria condivisa, ma, se già ne esiste uno con il medesimo flag, la funzione fallisce. Ciò è utile per motivi di sicurezza, infatti questo permette di evitare problemi di concorrenza.

Il terzo parametro, mode, indica i permessi che si desidera assegnare al segmento di memoria, questi sono i medesimi permessi utilizzati per un file. Occorre passare i permessi in forma ottale, ad esempio 0644. L'ultimo parametro è la dimensione in bytes del blocco di memoria condivisa che si desidera creare.

> **Nota:** Il terzo ed il quarto parametro dovrebbero essere a 0 se si sta aprendo un segmento di memoria esistene. Se la funzione **shmop_open()** ha successo, sarà restituito un id da usarsi per accedere al segmento di memoria condivisa appena creato.

Esempio 1. Creazione di un nuovo blocco di memoria condivisa

```
<?php
$shm_id = shmop_open(0x0fff, "c", 0644, 100);
?>
```

Questo esempio apre un blocco di memoria condivisa con id di sistema pari a 0x0fff.

shmop_read (PHP 4 >= 4.0.4)

Legge i dati da un segmento di memoria condivisa

string **shmop_read** (int shmid, int start, int count) \linebreak

La funzione **shmop_read()** legge una stringa da un blocco di memoria condivisa.

La funzione **shmop_read()** utilizza 3 parametri: shmid, che è l'identificativo del blocco di memoria condivisa creato da shmop_open(); start, che indica l'offset da cui partire a leggere e count che indica il numero dei byte da leggere.

Esempio 1. Lettura di un segmento di memoria condivisa

```
<?php
$shm_data = shmop_read($shm_id, 0, 50);
?>
```

Questo esempio legge 50 byte da un blocco di memoria condivisa e posiziona i dati nella variabile `$shm_data`.

shmop_size (PHP 4 >= 4.0.4)

Restituisce la dimensione di un blocco di memoria condivisa

int **shmop_size** (int shmid) \linebreak

Si utilizza la funzione **shmop_size()** per ottenere la dimensione in byte del segmento di memoria condivisa.

La funzione **shmop_size()** ha un solo parametro, shmid, che è l'identificativo del blocco di memoria condiviso creato da shmop_open(); la funzione restituisce un numero intero che rappresenta il numero dei byte occupati dal segmento di memoria condivisa.

Esempio 1. Come ottenere la dimensione della memoria condivisa

```
<?php
$shm_size = shmop_size($shm_id);
?>
```

In questo esempio si memorizza nella variabile `$shm_size` la dimensione del blocco di memoria identificato da `$shm_id`.

shmop_write (PHP 4 >= 4.0.4)

Scrittura di dati nel blocco di memoria condivisa

int **shmop_write** (int shmid, string data, int offset) \linebreak

La funzione **shmop_write()** scrive una stringa in un segmento di memoria condivisa.

La funzione **shmop_write()** utilizza 3 parametri: shmid, che è l'identificativo del blocco di memoria condiviso creato da shmop_open(); data, che è la stringa che si vuole scrivere nel blocco di memoria e offset, che specifica dove cominciare a scrivere nella memoria condivisa.

Esempio 1. Scrittura di un blocco di memoria condivisa

```
<?php
$shm_bytes_written = shmop_write($shm_id, $my_string, 0);
?>
```

Questo esempio scrive i dati della variabile `$my_string` nel blocco di memoria condivisa, mentre `$shm_bytes_written` contiene il numero dei byte scritti.

XCIII. Shockwave Flash functions

PHP offers the ability to create Shockwave Flash files via Paul Haeberli's libswf module. You can download libswf at ftp://ftp.sgi.com/sgi/graphics/grafica/flash. Once you have libswf all you need to do is to configure `--with-swf[=DIR]` where DIR is a location containing the directories include and lib. The include directory has to contain the swf.h file and the lib directory has to contain the libswf.a file. If you unpack the libswf distribution the two files will be in one directory. Consequently you will have to copy the files to the proper location manually.

Once you've successfully installed PHP with Shockwave Flash support you can then go about creating Shockwave files from PHP. You would be surprised at what you can do, take the following code:

Esempio 1. SWF example

```
<?php
swf_openfile ("test.swf", 256, 256, 30, 1, 1, 1);
swf_ortho2 (-100, 100, -100, 100);
swf_defineline (1, -70, 0, 70, 0, .2);
swf_definerect (4, 60, -10, 70, 0, 0);
swf_definerect (5, -60, 0, -70, 10, 0);
swf_addcolor (0, 0, 0, 0);

swf_definefont (10, "Mod");
swf_fontsize (5);
swf_fontslant (10);
swf_definetext (11, "This be Flash wit PHP!", 1);

swf_pushmatrix ();
swf_translate (-50, 80, 0);
swf_placeobject (11, 60);
swf_popmatrix ();

for ($i = 0; $i < 30; $i++) {
    $p = $i/(30-1);
    swf_pushmatrix ();
    swf_scale (1-($p*.9), 1, 1);
    swf_rotate (60*$p,  'z');
    swf_translate (20+20*$p, $p/1.5, 0);
    swf_rotate (270*$p,  'z');
    swf_addcolor ($p, 0, $p/1.2, -$p);
    swf_placeobject (1, 50);
    swf_placeobject (4, 50);
    swf_placeobject (5, 50);
    swf_popmatrix ();
    swf_showframe ();
}

for ($i = 0; $i < 30; $i++) {
    swf_removeobject (50);
    if (($i%4) == 0) {
        swf_showframe ();
    }
}
```

```
swf_startdoaction ();
swf_actionstop ();
swf_enddoaction ();

swf_closefile ();
?>
```

Nota: SWF support was added in PHP 4 RC2.

The libswf does not have support for Windows. The development of that library has been stopped, and the source is not available to port it to another systems.

For up to date SWF support take a look at the MING functions.

swf_actiongeturl (PHP 4 >= 4.0.0)

Get a URL from a Shockwave Flash movie

void **swf_actiongeturl** (string url, string target) \linebreak

The **swf_actionGetUrl()** function gets the URL specified by the parameter *url* with the target *target*.

swf_actiongotoframe (PHP 4 >= 4.0.0)

Play a frame and then stop

void **swf_actiongotoframe** (int framenumber) \linebreak

The **swf_actionGotoFrame()** function will go to the frame specified by *framenumber*, play it, and then stop.

swf_actiongotolabel (PHP 4 >= 4.0.0)

Display a frame with the specified label

void **swf_actiongotolabel** (string label) \linebreak

The **swf_actionGotoLabel()** function displays the frame with the label given by the *label* parameter and then stops.

swf_actionnextframe (PHP 4 >= 4.0.0)

Go foward one frame

void **swf_actionnextframe** (void) \linebreak

Go foward one frame.

swf_actionplay (PHP 4 >= 4.0.0)

Start playing the flash movie from the current frame

void **swf_actionplay** (void) \linebreak

Start playing the flash movie from the current frame.

swf_actionprevframe (PHP 4 >= 4.0.0)

Go backwards one frame

void **swf_actionprevframe** (void) \linebreak

swf_actionsettarget (PHP 4 >= 4.0.0)

Set the context for actions

void **swf_actionsettarget** (string target) \linebreak

The **swf_actionSetTarget()** function sets the context for all actions. You can use this to control other flash movies that are currently playing.

swf_actionstop (PHP 4 >= 4.0.0)

Stop playing the flash movie at the current frame

void **swf_actionstop** (void) \linebreak

Stop playing the flash movie at the current frame.

swf_actiontogglequality (PHP 4 >= 4.0.0)

Toggle between low and high quality

void **swf_actiontogglequality** (void) \linebreak

Toggle the flash movie between high and low quality.

swf_actionwaitforframe (PHP 4 >= 4.0.0)

Skip actions if a frame has not been loaded

void **swf_actionwaitforframe** (int framenumber, int skipcount) \linebreak

The **swf_actionWaitForFrame()** function will check to see if the frame, specified by the *`framenumber`* parameter has been loaded, if not it will skip the number of actions specified by the *`skipcount`* parameter. This can be useful for "Loading..." type animations.

swf_addbuttonrecord (PHP 4 >= 4.0.0)

Controls location, appearance and active area of the current button

void **swf_addbuttonrecord** (int states, int shapeid, int depth) \linebreak

The **swf_addbuttonrecord()** function allows you to define the specifics of using a button. The first parameter, *states*, defines what states the button can have, these can be any or all of the following constants: BSHitTest, BSDown, BSOver or BSUp. The second parameter, the *shapeid* is the look of the button, this is usually the object id of the shape of the button. The *depth* parameter is the placement of the button in the current frame.

Esempio 1. swf_addbuttonrecord() function example

```
swf_startButton ($objid, TYPE_MENUBUTTON);
    swf_addButtonRecord (BSDown|BSOver, $buttonImageId, 340);
    swf_onCondition (MenuEnter);
        swf_actionGetUrl ("http://www.designmultimedia.com", "_level1");
    swf_onCondition (MenuExit);
        swf_actionGetUrl ("", "_level1");
swf_endButton ();
```

swf_addcolor (PHP 4 >= 4.0.0)

Set the global add color to the rgba value specified

void **swf_addcolor** (float r, float g, float b, float a) \linebreak

The **swf_addcolor()** function sets the global add color to the *rgba* color specified. This color is then used (implicitly) by the swf_placeobject(), swf_modifyobject() and the swf_addbuttonrecord() functions. The color of the object will be add by the *rgba* values when the object is written to the screen.

Nota: The *rgba* values can be either positive or negative.

swf_closefile (PHP 4 >= 4.0.0)

Close the current Shockwave Flash file

void **swf_closefile** ([int return_file]) \linebreak

Close a file that was opened by the swf_openfile() function. If the *return_file* parameter is set then the contents of the SWF file are returned from the function.

Esempio 1. Creating a simple flash file based on user input and outputting it and saving it in a database

```
<?php

// The $text variable is submitted by the
// user

// Global variables for database
// access (used in the swf_savedata() function)
$DBHOST = "localhost";
$DBUSER = "sterling";
$DBPASS = "secret";

swf_openfile ("php://stdout", 256, 256, 30, 1, 1, 1);

    swf_definefont (10, "Ligon-Bold");
        swf_fontsize (12);
        swf_fontslant (10);

    swf_definetext (11, $text, 1);

    swf_pushmatrix ();
        swf_translate (-50, 80, 0);
        swf_placeobject (11, 60);
    swf_popmatrix ();

    swf_showframe ();

    swf_startdoaction ();
        swf_actionstop ();
    swf_enddoaction ();

$data = swf_closefile (1);

$data ?
  swf_savedata ($data) :
  die ("Error could not save SWF file");

// void swf_savedata (string data)
// Save the generated file a database
// for later retrieval
function swf_savedata ($data)
{
    global $DBHOST,
           $DBUSER,
           $DBPASS;

    $dbh = @mysql_connect ($DBHOST, $DBUSER, $DBPASS);

    if (!$dbh) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    $stmt = "INSERT INTO swf_files (file) VALUES ('$data')";
```

```
    $sth = @mysql_query ($stmt, $dbh);

    if (!$sth) {
        die (sprintf ("Error [%d]: %s",
                      mysql_errno (), mysql_error ()));
    }

    @mysql_free_result ($sth);
    @mysql_close ($dbh);
}
?>
```

swf_definebitmap (PHP 4 >= 4.0.0)

Define a bitmap

void **swf_definebitmap** (int objid, string image_name) \linebreak

The **swf_definebitmap()** function defines a bitmap given a GIF, JPEG, RGB or FI image. The image will be converted into a Flash JPEG or Flash color map format.

swf_definefont (PHP 4 >= 4.0.0)

Defines a font

void **swf_definefont** (int fontid, string fontname) \linebreak

The **swf_definefont()** function defines a font given by the *fontname* parameter and gives it the id specified by the *fontid* parameter. It then sets the font given by *fontname* to the current font.

swf_defineline (PHP 4 >= 4.0.0)

Define a line

void **swf_defineline** (int objid, float x1, float y1, float x2, float y2, float width) \linebreak

The **swf_defineline()** defines a line starting from the x coordinate given by *x1* and the y coordinate given by *y1* parameter. Up to the x coordinate given by the *x2* parameter and the y coordinate given by the *y2* parameter. It will have a width defined by the *width* parameter.

swf_definepoly (PHP 4 >= 4.0.0)

Define a polygon

void **swf_definepoly** (int objid, array coords, int npoints, float width) \linebreak

The **swf_definepoly()** function defines a polygon given an array of x, y coordinates (the coordinates are defined in the parameter *coords*). The parameter *npoints* is the number of overall points that are contained in the array given by *coords*. The *width* is the width of the polygon's border, if set to 0.0 the polygon is filled.

swf_definerect (PHP 4 >= 4.0.0)

Define a rectangle

void **swf_definerect** (int objid, float x1, float y1, float x2, float y2, float width) \linebreak

The **swf_definerect()** defines a rectangle with an upper left hand coordinate given by the x, *x1*, and the y, *y1*. And a lower right hand coordinate given by the x coordinate, *x2*, and the y coordinate, *y2* . Width of the rectangles border is given by the *width* parameter, if the width is 0.0 then the rectangle is filled.

swf_definetext (PHP 4 >= 4.0.0)

Define a text string

void **swf_definetext** (int objid, string str, int docenter) \linebreak

Define a text string (the *str* parameter) using the current font and font size. The *docenter* is where the word is centered, if *docenter* is 1, then the word is centered in x.

swf_endbutton (PHP 4 >= 4.0.0)

End the definition of the current button

void **swf_endbutton** (void) \linebreak

The **swf_endButton()** function ends the definition of the current button.

swf_enddoaction (PHP 4 >= 4.0.0)

End the current action

void **swf_enddoaction** (void) \linebreak

Ends the current action started by the swf_startdoaction() function.

swf_endshape (PHP 4 >= 4.0.0)

Completes the definition of the current shape

void **swf_endshape** (void) \linebreak

The **swf_endshape()** completes the definition of the current shape.

swf_endsymbol (PHP 4 >= 4.0.0)

End the definition of a symbol

void **swf_endsymbol** (void) \linebreak

The **swf_endsymbol()** function ends the definition of a symbol that was started by the swf_startsymbol() function.

swf_fontsize (PHP 4 >= 4.0.0)

Change the font size

void **swf_fontsize** (float size) \linebreak

The **swf_fontsize()** function changes the font size to the value given by the `size` parameter.

swf_fontslant (PHP 4 >= 4.0.0)

Set the font slant

void **swf_fontslant** (float slant) \linebreak

Set the current font slant to the angle indicated by the `slant` parameter. Positive values create a foward slant, negative values create a negative slant.

swf_fonttracking (PHP 4 >= 4.0.0)

Set the current font tracking

void **swf_fonttracking** (float tracking) \linebreak

Set the font tracking to the value specified by the `tracking` parameter. This function is used to increase the spacing between letters and text, positive values increase the space and negative values decrease the space between letters.

swf_getbitmapinfo (PHP 4 >= 4.0.0)

Get information about a bitmap

array **swf_getbitmapinfo** (int bitmapid) \linebreak

The **swf_getbitmapinfo()** function returns an array of information about a bitmap given by the *bitmapid* parameter. The returned array has the following elements:

- "size" - The size in bytes of the bitmap.
- "width" - The width in pixels of the bitmap.
- "height" - The height in pixels of the bitmap.

swf_getfontinfo (PHP 4 >= 4.0.0)

The height in pixels of a capital A and a lowercase x

array **swf_getfontinfo** (void) \linebreak

The **swf_getfontinfo()** function returns an associative array with the following parameters:

- Aheight - The height in pixels of a capital A.
- xheight - The height in pixels of a lowercase x.

swf_getframe (PHP 4 >= 4.0.0)

Get the frame number of the current frame

int **swf_getframe** (void) \linebreak

The **swf_getframe()** function gets the number of the current frame.

swf_labelframe (PHP 4 >= 4.0.0)

Label the current frame

void **swf_labelframe** (string name) \linebreak

Label the current frame with the name given by the *name* parameter.

swf_lookat (PHP 4 >= 4.0.0)

Define a viewing transformation

void **swf_lookat** (float view_x, float view_y, float view_z, float reference_x, float reference_y, float reference_z, float twist) \linebreak

The **swf_lookat()** function defines a viewing transformation by giving the viewing position (the parameters *view_x*, *view_y*, and *view_z*) and the coordinates of a reference point in the scene, the reference point is defined by the *reference_x*, *reference_y* , and *reference_z* parameters. The *twist* controls the rotation along with viewer's z axis.

swf_modifyobject (PHP 4 >= 4.0.0)

Modify an object

void **swf_modifyobject** (int depth, int how) \linebreak

Updates the position and/or color of the object at the specified depth, *depth*. The parameter *how* determines what is updated. *how* can either be the constant MOD_MATRIX or MOD_COLOR or it can be a combination of both (MOD_MATRIX|MOD_COLOR).

MOD_COLOR uses the current mulcolor (specified by the function swf_mulcolor()) and addcolor (specified by the function swf_addcolor()) to color the object. MOD_MATRIX uses the current matrix to position the object.

swf_mulcolor (PHP 4 >= 4.0.0)

Sets the global multiply color to the rgba value specified

void **swf_mulcolor** (float r, float g, float b, float a) \linebreak

The **swf_mulcolor()** function sets the global multiply color to the *rgba* color specified. This color is then used (implicitly) by the swf_placeobject(), swf_modifyobject() and the swf_addbuttonrecord() functions. The color of the object will be multiplied by the *rgba* values when the object is written to the screen.

Nota: The *rgba* values can be either positive or negative.

swf_nextid (PHP 4 >= 4.0.0)

Returns the next free object id

int **swf_nextid** (void) \linebreak

The **swf_nextid()** function returns the next available object id.

swf_oncondition (PHP 4 >= 4.0.0)

Describe a transition used to trigger an action list

void **swf_oncondition** (int transition) \linebreak

The **swf_onCondition()** function describes a transition that will trigger an action list. There are several types of possible transitions, the following are for buttons defined as TYPE_MENUBUTTON:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- IdletoOverDown
- OutDowntoIdle
- MenuEnter (IdletoOverUp|IdletoOverDown)
- MenuExit (OverUptoIdle|OverDowntoIdle)

For TYPE_PUSHBUTTON there are the following options:

- IdletoOverUp
- OverUptoIdle
- OverUptoOverDown
- OverDowntoOverUp
- OverDowntoOutDown
- OutDowntoOverDown
- OutDowntoIdle
- ButtonEnter (IdletoOverUp|OutDowntoOverDown)
- ButtonExit (OverUptoIdle|OverDowntoOutDown)

swf_openfile (PHP 4 >= 4.0.0)

Open a new Shockwave Flash file

void **swf_openfile** (string filename, float width, float height, float framerate, float r, float g, float b) \linebreak

The **swf_openfile()** function opens a new file named *filename* with a width of *width* and a height of *height* a frame rate of *framerate* and background with a red color of *r* a green color of *g* and a blue color of *b*.

The **swf_openfile()** must be the first function you call, otherwise your script will cause a segfault. If you want to send your output to the screen make the filename: "php://stdout" (support for this is in 4.0.1 and up).

swf_ortho (PHP 4)

Defines an orthographic mapping of user coordinates onto the current viewport

void **swf_ortho** (float xmin, float xmax, float ymin, float ymax, float zmin, float zmax) \linebreak

The **swf_ortho()** function defines a orthographic mapping of user coordinates onto the current viewport.

swf_ortho2 (PHP 4 >= 4.0.0)

Defines 2D orthographic mapping of user coordinates onto the current viewport

void **swf_ortho2** (float xmin, float xmax, float ymin, float ymax) \linebreak

The **swf_ortho2()** function defines a two dimensional orthographic mapping of user coordinates onto the current viewport, this defaults to one to one mapping of the area of the Flash movie. If a perspective transformation is desired, the **swf_perspective ()** function can be used.

swf_perspective (PHP 4 >= 4.0.0)

Define a perspective projection transformation

void **swf_perspective** (float fovy, float aspect, float near, float far) \linebreak

The **swf_perspective()** function defines a perspective projection transformation. The `fovy` parameter is field-of-view angle in the y direction. The `aspect` parameter should be set to the aspect ratio of the viewport that is being drawn onto. The `near` parameter is the near clipping plane and the `far` parameter is the far clipping plane.

> **Nota:** Various distortion artifacts may appear when performing a perspective projection, this is because Flash players only have a two dimensional matrix. Some are not to pretty.

swf_placeobject (PHP 4 >= 4.0.0)

Place an object onto the screen

void **swf_placeobject** (int objid, int depth) \linebreak

Places the object specified by `objid` in the current frame at a depth of `depth`. The `objid` parameter and the `depth` must be between 1 and 65535.

This uses the current mulcolor (specified by swf_mulcolor()) and the current addcolor (specified by swf_addcolor()) to color the object and it uses the current matrix to position the object.

> **Nota:** Full RGBA colors are supported.

swf_polarview (PHP 4 >= 4.0.0)

Define the viewer's position with polar coordinates

void **swf_polarview** (float dist, float azimuth, float incidence, float twist) \linebreak

The **swf_polarview()** function defines the viewer's position in polar coordinates. The *dist* parameter gives the distance between the viewpoint to the world space origin. The *azimuth* parameter defines the azimuthal angle in the x,y coordinate plane, measured in distance from the y axis. The *incidence* parameter defines the angle of incidence in the y,z plane, measured in distance from the z axis. The incidence angle is defined as the angle of the viewport relative to the z axis. Finally the *twist* specifies the amount that the viewpoint is to be rotated about the line of sight using the right hand rule.

swf_popmatrix (PHP 4 >= 4.0.0)

Restore a previous transformation matrix

void **swf_popmatrix** (void) \linebreak

The **swf_popmatrix()** function pushes the current transformation matrix back onto the stack.

swf_posround (PHP 4 >= 4.0.0)

Enables or Disables the rounding of the translation when objects are placed or moved

void **swf_posround** (int round) \linebreak

The **swf_posround()** function enables or disables the rounding of the translation when objects are placed or moved, there are times when text becomes more readable because rounding has been enabled. The *round* is whether to enable rounding or not, if set to the value of 1, then rounding is enabled, if set to 0 then rounding is disabled.

swf_pushmatrix (PHP 4 >= 4.0.0)

Push the current transformation matrix back unto the stack

void **swf_pushmatrix** (void) \linebreak

The **swf_pushmatrix()** function pushes the current transformation matrix back onto the stack.

swf_removeobject (PHP 4 >= 4.0.0)

Remove an object

void **swf_removeobject** (int depth) \linebreak

Removes the object at the depth specified by *depth*.

swf_rotate (PHP 4 >= 4.0.0)

Rotate the current transformation

void **swf_rotate** (float angle, string axis) \linebreak

The **swf_rotate()** rotates the current transformation by the angle given by the *angle* parameter around the axis given by the *axis* parameter. Valid values for the axis are 'x' (the x axis), 'y' (the y axis) or 'z' (the z axis).

swf_scale (PHP 4 >= 4.0.0)

Scale the current transformation

void **swf_scale** (float x, float y, float z) \linebreak

The **swf_scale()** scales the x coordinate of the curve by the value of the *x* parameter, the y coordinate of the curve by the value of the *y* parameter, and the z coordinate of the curve by the value of the *z* parameter.

swf_setfont (PHP 4 >= 4.0.0)

Change the current font

void **swf_setfont** (int fontid) \linebreak

The **swf_setfont()** sets the current font to the value given by the *fontid* parameter.

swf_setframe (PHP 4 >= 4.0.0)

Switch to a specified frame

void **swf_setframe** (int framenumber) \linebreak

The **swf_setframe()** changes the active frame to the frame specified by *framenumber*.

swf_shapearc (PHP 4 >= 4.0.0)

Draw a circular arc

void **swf_shapearc** (float x, float y, float r, float ang1, float ang2) \linebreak

The **swf_shapeArc()** function draws a circular arc from angle A given by the *ang1* parameter to angle B given by the *ang2* parameter. The center of the circle has an x coordinate given by the *x* parameter and a y coordinate given by the *y*, the radius of the circle is given by the *r* parameter.

swf_shapecurveto (PHP 4 >= 4.0.0)

Draw a quadratic bezier curve between two points

void **swf_shapecurveto** (float x1, float y1, float x2, float y2) \linebreak

The **swf_shapecurveto()** function draws a quadratic bezier curve from the current location, though the x coordinate given by *x1* and the y coordinate given by *y1* to the x coordinate given by *x2* and the y coordinate given by *y2*. The current position is then set to the x,y coordinates given by the *x2* and *y2* parameters

swf_shapecurveto3 (PHP 4 >= 4.0.0)

Draw a cubic bezier curve

void **swf_shapecurveto3** (float x1, float y1, float x2, float y2, float x3, float y3) \linebreak

Draw a cubic bezier curve using the x,y coordinate pairs *x1*, *y1* and *x2*,*y2* as off curve control points and the x,y coordinate *x3*, *y3* as an endpoint. The current position is then set to the x,y coordinate pair given by *x3*,*y3*.

swf_shapefillbitmapclip (PHP 4 >= 4.0.0)

Set current fill mode to clipped bitmap

void **swf_shapefillbitmapclip** (int bitmapid) \linebreak

Sets the fill to bitmap clipped, empty spaces will be filled by the bitmap given by the *bitmapid* parameter.

swf_shapefillbitmaptile (PHP 4 >= 4.0.0)

Set current fill mode to tiled bitmap

void **swf_shapefillbitmaptile** (int bitmapid) \linebreak

Sets the fill to bitmap tile, empty spaces will be filled by the bitmap given by the *bitmapid* parameter (tiled).

swf_shapefilloff (PHP 4 >= 4.0.0)

Turns off filling

void **swf_shapefilloff** (void) \linebreak

The **swf_shapeFillOff()** function turns off filling for the current shape.

swf_shapefillsolid (PHP 4 >= 4.0.0)

Set the current fill style to the specified color

void **swf_shapefillsolid** (float r, float g, float b, float a) \linebreak

The **swf_shapeFillSolid()** function sets the current fill style to solid, and then sets the fill color to the values of the *rgba* parameters.

swf_shapelinesolid (PHP 4 >= 4.0.0)

Set the current line style

void **swf_shapelinesolid** (float r, float g, float b, float a, float width) \linebreak

The **swf_shapeLineSolid()** function sets the current line style to the color of the *rgba* parameters and width to the *width* parameter. If 0.0 is given as a width then no lines are drawn.

swf_shapelineto (PHP 4 >= 4.0.0)

Draw a line

void **swf_shapelineto** (float x, float y) \linebreak

The **swf_shapeLineTo()** draws a line to the x,y coordinates given by the *x* parameter & the *y* parameter. The current position is then set to the x,y parameters.

swf_shapemoveto (PHP 4 >= 4.0.0)

Move the current position

void **swf_shapemoveto** (float x, float y) \linebreak

The **swf_shapeMoveTo()** function moves the current position to the x coordinate given by the *x* parameter and the y position given by the *y* parameter.

swf_showframe (PHP 4 >= 4.0.0)

Display the current frame

void **swf_showframe** (void) \linebreak

The swf_showframe function will output the current frame.

swf_startbutton (PHP 4 >= 4.0.0)

Start the definition of a button

void **swf_startbutton** (int objid, int type) \linebreak

The **swf_startbutton()** function starts off the definition of a button. The *type* parameter can either be TYPE_MENUBUTTON or TYPE_PUSHBUTTON. The TYPE_MENUBUTTON constant allows the focus to travel from the button when the mouse is down, TYPE_PUSHBUTTON does not allow the focus to travel when the mouse is down.

swf_startdoaction (PHP 4 >= 4.0.0)

Start a description of an action list for the current frame

void **swf_startdoaction** (void) \linebreak

The **swf_startdoaction()** function starts the description of an action list for the current frame. This must be called before actions are defined for the current frame.

swf_startshape (PHP 4 >= 4.0.0)

Start a complex shape

void **swf_startshape** (int objid) \linebreak

The **swf_startshape()** function starts a complex shape, with an object id given by the *objid* parameter.

swf_startsymbol (PHP 4 >= 4.0.0)

Define a symbol

void **swf_startsymbol** (int objid) \linebreak

Define an object id as a symbol. Symbols are tiny flash movies that can be played simultaneously. The *objid* parameter is the object id you want to define as a symbol.

swf_textwidth (PHP 4 >= 4.0.0)

Get the width of a string

float **swf_textwidth** (string str) \linebreak

The **swf_textwidth()** function gives the width of the string, *str*, in pixels, using the current font and font size.

swf_translate (PHP 4 >= 4.0.0)

Translate the current transformations

void **swf_translate** (float x, float y, float z) \linebreak

The **swf_translate()** function translates the current transformation by the *x*, *y*, and *z* values given.

swf_viewport (PHP 4 >= 4.0.0)

Select an area for future drawing

void **swf_viewport** (float xmin, float xmax, float ymin, float ymax) \linebreak

The **swf_viewport()** function selects an area for future drawing for *xmin* to *xmax* and *ymin* to *ymax*, if this function is not called the area defaults to the size of the screen.

XCIV. Funzioni per SNMP

Per potere utilizzare le funzioni SNMP su un sistema Unix, occorre installare il pacchetto UCD SNMP (http://net-snmp.sourceforge.net/). Sui sistemi Windows, invece, le funzioni SNMP sono disponibili soltanto su NT e non su sistemi Windows 95 e 98.

Attenzione: per potere usare il pacchetto UCD SNMP, occorre definire NO_ZEROLENGTH_COMMUNITY a 1 prima di compilarlo. Dopo avere configurato UCD SNMP, occorre editare il file config.h, cercare NO_ZEROLENGTH_COMMUNITY e decommentare la linea #define. Alla fine si deve ottenere:

```
#define NO_ZEROLENGTH_COMMUNITY 1
```

Se durante l'uso dei comandi SNMP dovessero comparire degli errori di "segmentation fault", non seguire le istruzioni precedenti. Se non si desidera ricompilare il pacchetto UCD SNMP, si può optare per compilare PHP con l'opzione --enable-ucd-snmp-hack che aggira questo problema.

snmp_get_quick_print (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Restituisce il valore corrente per il parametro quick_print della libreria UCD

bool **snmp_get_quick_print** (void) \linebreak

La funzione restituisce il valore del parametro quick_print della libreria UCD. Per default quick_print è settato ad off.

```
$quickprint = snmp_get_quick_print();
```

Nell'esempio precedente la funzione restituirebbe `FALSE` se quick_print fosse ad off, `TRUE` se quick_print fosse ad on.

La funzione **snmp_get_quick_print()** è disponibile soltanto con l'uso della libreria UCD SNMP. Questa funzione non è disponibile nella libreria SNMP per Windows.

Vedere snmp_set_quick_print() per una descrizione completa di quick_print.

snmp_set_quick_print (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Setta il valore di quick_print

void **snmp_set_quick_print** (bool quick_print) \linebreak

La funzione setta il valore del parametro quick_print della libreria UCD SNMP. Quando è attivo (1), la libreria SNMP restituisce valori 'quick printed'. Ciò significa che saranno visualizzati solo i valori. Quando quick_print non è attivo (default), la libreria UCD SNMP visualizzerà informazioni extra tra i quali il tipo del valore (per esempio IpAddress oppure OID). Inoltre, se quick_print non è abilitato, la libreria visualizza il valore esadecimale per tutte le stringhe di tre caratteri o meno.

L'attivazione di quick_print viene spesso usata quando l'informazione restuita viene utilizzata piuttosto che visualizzata.

```
snmp_set_quick_print(0);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
snmp_set_quick_print(1);
$a = snmpget("127.0.0.1", "public", ".1.3.6.1.2.1.2.2.1.9.1");
echo "$a<BR>\n";
```

Il primo valore visualizzato può essere 'Timeticks: (0) 0:00:00.00', mentre con quick_print abilitato sarebbe stato '0:00:00.00'.

Per default la libreria UCD SNMP restituisce valori discorsivi, mentre quick_print viene usato per avere solo il valore.

Attualmente le stringhe sono restituite con apici aggiuntivi, questo sarà corretto in una release successiva.

La funzione **snmp_set_quick_print()** è disponibile soltanto con l'uso della libreria UCD SNMP. Questa funzione non è disponibile nella libreria SNMP per Windows.

snmpget (PHP 3, PHP 4 >= 4.0.0)

Preleva un oggetto SNMP

string **snmpget** (string hostname, string community, string object_id [, int timeout [, int retries]]) \linebreak

La funzione restituisce il valore di un oggetto SNMP se ha successo, altrimenti `FALSE` se si verificano errori

La funzione **snmpget()**, viene utilizzata per leggere il valore dell'oggetto SNMP specificato da *`object_id`*. L'agente SNMP a cui accedere viene specificato nel parametro *`hostname`*, mentre la comunità viene indicata in *`community`*.

```
$syscontact = snmpget("127.0.0.1", "public", "system.SysContact.0");
```

snmprealwalk (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Restituisce tutti gli oggetti compresi i rispettivi id

array **snmprealwalk** (string host, string community, string object_id [, int timeout [, int retries]]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

snmpset (PHP 3>= 3.0.12, PHP 4 >= 4.0.0)

Valorizza un oggetto SNMP

bool **snmpset** (string hostname, string community, string object_id, string type, mixed value [, int timeout [, int retries]]) \linebreak

Setta il valore di un specifico oggetto SNMP. La funzione restituisce `TRUE` se ha successo, `FALSE` se si verifica un errore.

La funzione **snmpset()** viene usata per settare il valore dell'oggetto SNMP indicato dal parametro *object_id*. L'agente SNMP viene indicato nel parametro *hostname* e la comunità viene specificata nel parametro *community*.

snmpwalk (PHP 3, PHP 4 >= 4.0.0)

Scarica tutti gli oggetti SNMP da un agente

array **snmpwalk** (string hostname, string community, string object_id [, int timeout [, int retries]]) \linebreak

La funzione restituisce un array con i valori degli oggetti SNMP utilizzando *object_id* come punto di partenza, oppure `FALSE` se si verifica un errore.

La funzione **snmpwalk()** viene utilizzata per leggere tutti i valori dall'agente SNMP specificato nel parametro *hostname*. Il parametro *Community* specifica la comunità per l'agente. Con l'impostazione a `NULL` del parametro *object_id* si indica la radice dell'albero degli oggetti SNMP, pertanto saranno restituiti nell'array tutti gli oggetti dell'albero. Viceversa se si indica un valore per *object_id*, sarranno restituiti tutti gli oggetti sottostanti *object_id*.

```
$a = snmpwalk("127.0.0.1", "public", "");
```

L'esempio precedente mostra come recuperare tutti gli oggetti SNMP dall'agente attivo sulla macchina locale. Tramite un loop (illustrato di seguito) si può accedere a tutti i valori.

```
for ($i=0; $i < count($a); $i++) {
    echo $a[$i];
}
```

snmpwalkoid (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Richiesta dell'albero delle informazioni di una macchina di rete

array **snmpwalkoid** (string hostname, string community, string object_id [, int timeout [, int retries]]) \line-break

La funzione restituisce un array associativo contenente gli id degli oggetti ed il loro rispettivo valore usando l'oggetto indicato in *object_id* come radice. Se si verificano degli errori la funzione restituisce `FALSE`.

La funzione **snmpwalkoid()** viene utilizzata per leggere gli id di tutti gli oggetti SNMP ed i relativi valori da un agente SNMP presente sul server indicato da *hostname*. La comunità viene specificata nel parametro *community*. Con l'impostazione a `NULL` del parametro *object_id* si indica la

radice dell'albero degli oggetti SNMP, pertanto saranno restituiti nell'array tutti gli oggetti dell'albero. Viceversa se si indica un valore per *object_id*, sarranno restituiti tutti gli oggetti sottostanti a *object_id*.

La presenza delle due funzioni **snmpwalkoid()** e snmpwalk() ha ragioni storiche. Sono presenti entrambe per compatibilità con il passato.

```
$a = snmpwalkoid("127.0.0.1", "public", "");
```

L'esempio precedente mostra come recuperare tutti gli oggetti SNMP dall'agente attivo sulla macchina locale. Tramite un loop (illustrato di seguito) si può accedere a tutti i valori.

```
for (reset($a); $i = key($a); next($a)) {
    echo "$i: $a[$i]<br>\n";
}
```

XCV. Funzioni relative ai Socket

> **Attenzione**
>
> Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

Questa estensione implementa una interfaccia a basso livello verso i socket, fornendo la possibilità di agire sia come server sia come client.

Le funzioni relative ai socket qui descritte, sono parte di una estensione di PHP che deve essere abilitata durante la fase di compila tramite l'opzione `--enable-sockets` di **configure**.

Per un esempio di una interfaccia generica lato client, vedere fsockopen() e pfsockopen().

Per l'utilizzo di queste funzioni, è importante ricordare che molte di esse hanno il medesimo nome della loro controparte in C, ma spesso hanno dichiarazioni differenti. Ricordarsi di leggere la descrizione per evitare confusione.

Per chi non ha famigliarità con la programmazione dei socket, può trovare utili informazioni nelle pagine del manuale di Unix, ed inoltre sul web si può trovare diversi tutorial sulla programmazione dei socket in C, molti dei quali possono essere utilizzati, con lievi modifiche, nella programmazione dei socket in PHP.

Esempio 1. Esempio di programma con i socket: semplice server TCP/IP

Questo esempio mostra un semplice server. Occorre modificare le variabili `address` e `port` per adeguarle ai parametri della macchina su cui sarà eseguito. Ci si può connettere al server con un comando simile a **telnet 192.168.1.53 10000** (dove l'indirizzo e la porta devono essere uguali a quanto indicato nel setup). Qualsiasi lettara sarà digitata, verrà visualizzata sul server e sul client. Per disconnettersi digitare 'quit'.

```
#!/usr/local/bin/php -q
<?php
error_reporting (E_ALL);

/* Si indica allo script di non uscire mentre attende una connessione */
set_time_limit (0);

/* Abilita lo scarico dell'output così si è in grado di vedere cosa passa
 * non appena arrivano i dati al server. */
ob_implicit_flush ();

$address = '192.168.1.53';
$port = 10000;

if (($sock = socket_create (AF_INET, SOCK_STREAM, 0)) < 0) {
    echo "socket_create() fallito: motivo: " . socket_strerror ($sock) . "\n";
}

if (($ret = socket_bind ($sock, $address, $port)) < 0) {
    echo "socket_bind() fallito: motivo: " . socket_strerror ($ret) . "\n";
}
```

```
if (($ret = socket_listen ($sock, 5)) < 0) {
    echo "socket_listen() fallito: motivo: " . socket_strerror ($ret) . "\n";
}

do {
    if (($msgsock = socket_accept($sock)) < 0) {
        echo "socket_accept() fallito: motivo: " . socket_strerror ($msgsock) . "\n";
        break;
    }
    /* Invio delle istruzioni */
    $msg = "\nBenvenuti al server di test in PHP. \n" .
        "Per uscire, digitare ’quit’. Per chiudere il server digitare ’shutdown’.\n";
    socket_write($msgsock, $msg, strlen($msg));

    do {
        if (FALSE === ($buf = socket_read ($msgsock, 2048))) {
            echo "socket_read() fallito: motivo: " . socket_strerror ($ret) . "\n";
            break 2;
        }
        if (!$buf = trim ($buf)) {
            continue;
        }
        if ($buf == ’quit’) {
            break;
        }
        if ($buf == ’shutdown’) {
            socket_close ($msgsock);
            break 2;
        }
        $talkback = "PHP: Testo scritto ’$buf’.\n";
        socket_write ($msgsock, $talkback, strlen ($talkback));
        echo "$buf\n";
    } while (true);
    socket_close ($msgsock);
} while (true);

socket_close ($sock);
?>
```

Esempio 2. Esempio di programma con i socket: semplice client TCP/IP

In questo esempio sarà illustrato un semplice client HTTP. Questo, molto semplicemente, si collega ad un server, invia una richiesta HEAD, visualizza la risposta ed esce.

```
<?php
error_reporting (E_ALL);

echo "<h2>Connessione TCP/IP </h2>\n";

/* Ottiene la porta per il servizio WWW. */
$service_port = getservbyname (’www’, ’tcp’);
```

```
/* Ottiene l’indirizzo IP per il server cercato. */
$address = gethostbyname (’www.php.net’);

/* Crea un socket TCP/IP. */
$socket = socket_create (AF_INET, SOCK_STREAM, 0);
if ($socket < 0) {
    echo "socket_create() fallito: motivo: " . socket_strerror ($socket) . "\n";
} else {
    echo "OK.\n";
}

echo "Tentativo di connessione a ’$address’ sulla porta ’$service_port’...";
$result = socket_connect ($socket, $address, $service_port);
if ($result < 0) {
    echo "socket_connect() fallito.\nMotivo: ($result) " . socket_strerror($result) . "\n
} else {
    echo "OK.\n";
}

$in = "HEAD / HTTP/1.0\r\n\r\n";
$out = ’’;

echo "Invio HTTP HEAD...";
socket_write ($socket, $in, strlen ($in));
echo "OK.\n";

echo "Lettura della risposta:\n\n";
while ($out = socket_read ($socket, 2048)) {
    echo $out;
}

echo "Chiusura del socket...";
socket_close ($socket);
echo "OK.\n\n";
?>
```

socket_accept (PHP 4 >= 4.1.0)

Accetta una connessione su un socket

int **socket_accept** (resource socket) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Dopo la creazione del socket *socket* con socket_create(), l'assegnazione di un nome con socket_bind(), e averlo messo in attesa di connessione con socket_listen(), con questa funzione si inizia ad accettare le richieste di connessione su quel socket. Una volta che si ha una connessione, la funzione restituisce un nuovo socket che può essere usato per la comunicazione. Se vi sono diverse richieste di connessioni pendenti verrà utilizzata la prima. Viceversa se non vi sono richieste in attesa la funzione **socket_accept()** si blocca in attesa di una richiesta. Se il *socket* è stato configurato "non-blocking" con socket_set_blocking() o con socket_set_nonblock(), la funzione restituirà un errore.

Il descrittore di socket restituito da **socket_accept()** non può essere utilizzato per acquisire nuove connesioni. Per questo scopo occorre continuare ad usare il socket originale, indicato in *socket*, che rimane aperto.

La funzione restituisce un descrittore di socket in caso di esito positivo, in caso di errore si avrà un codice di errore negativo. Questo codice può essere passato a socket_strerror() per ottenere una descrizione dell'errore.

Vedere anche socket_bind(), socket_connect(), socket_listen(), socket_create(), socket_get_status() e socket_strerror().

socket_bind (PHP 4 >= 4.1.0)

Bind di un nome ad un socket

int **socket_bind** (resource socket, string address [, int port]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

La funzione **socket_bind()** esegue il bind del nome passato in *address* sul socket indicato da *socket*, che deve essere un descrittore valido creato da socket_create().

Il parametro *address* può essere sia un classico indirizzo IP (ad esempio `127.0.0.1`), se il socket appartiene alla famiglia `AF_INET`, sia il percorso di un socket nel dominio Unix, se il socket appartiene alla famiglia `AF_UNIX`.

Il parametro *port*, si utilizza soltanto con le connessioni tramite un socket di tipo `AF_INET`, ed indica quale porta sul server remoto si debba utilizzare per eseguire la connessione.

La funzione restituisce zero se ha successo, oppure un codice di errore negativo su errore. Questo codice può essere passato alla funzione socket_strerror() per ottenere una descrizione dell'errore.

Vedere anche socket_connect(), socket_listen(), socket_create(), socket_get_status() e socket_strerror().

socket_clear_error (PHP 4 >= 4.2.0)

Clears the error on the socket or the last error code

void **socket_clear_error** ([resource socket]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

This function clears the error code on the given socket or the global last socket error.

This function allows explicitely resetting the error code value either of a socket or of the extension global last error code. This may be useful to detect within a part of the application if an error occured or not.

See also socket_last_error() and socket_strerror().

socket_close (PHP 4 >= 4.1.0)

Chiude il descrittore di un socket

bool **socket_close** (resource socket) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

La funzione **socket_close()** chiude il descrittore di file (o di socket) indicato nel parametro *socket*.

Si noti che la funzione **socket_close()** non dovrebbe essere applicata ai descrittori di file di PHP creati con fopen(), popen(), fsockopen(), oppure **psockopen()**; ma soltanto con i socket creati dalle funzioni socket_create() e socket_accept().

La funzione ritorna `TRUE` se ha successo, oppure `FALSE` se si ha un errore (ad esempio,il *socket* non è valido).

Vedere anche socket_bind(), socket_listen(), socket_create(), socket_get_status() e socket_strerror().

socket_connect (PHP 4 >= 4.1.0)

Inizia una connessione su un socket

bool **socket_connect** (resource socket, string address [, int port]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Apre una connessione usando il descrittore di socket *socket*, il quale deve essere un descrittore di socket valido generato da socket_create().

Il parametro *address* può essere sia un classico indirizzo IP (ad esempio `127.0.0.1`), se il socket appartiene alla famiglia `AF_INET`, sia il percorso di un socket nel dominio Unix, se il socket appartiene alla famiglia `AF_UNIX`.

Il parametro *port*, utilizzato soltanto con le connessioni tramite un socket di tipo `AF_INET`, indica quale porta sul server remoto si debba utilizzare per eseguire la connessione.

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento. Questo codice può essere passato alla funzione socket_strerror() per ottenere una descrizione dell'errore.

Vedere anche socket_bind(), socket_listen(), socket_create(), socket_get_status() e socket_strerror().

socket_create (PHP 4 >= 4.1.0)

Crea un socket (punto terminale di una comunicazione).

resource **socket_create** (int domain, int type, int protocol) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

La funzione crea un punto terminale di una comunicazione (un socket) e restituisce un descrittore del socket.

Il parametro *domain* indica il dominio. Attualmente i valori ammessi sono `AF_INET` e `AF_UNIX`.

Il parametro *type* indica il tipo di socket. I tipi ammessi sono `SOCK_STREAM`, `SOCK_DGRAM`, `SOCK_SEQPACKET`, `SOCK_RAW`, `SOCK_RDM` oppure `SOCK_PACKET`.

Il parametro *protocol* indica il protocollo.

La funzione restituisce un descrittore di socket se ha successo, oppure un codice di errore negativo in caso di errore. Questo codice può essere passato alla funzione socket_strerror() per ottenere una descrizione dell'errore.

Per maggiori dettagli sull'utilizzo di **socket_create()**, e sul significato dei vari parametri si può consultare il man di Unix alle pagine socket (2).

Vedere anche socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_strerror() e socket_get_status().

socket_create_listen (PHP 4 >= 4.1.0)

Apre un socket per accettare connessioni su una porta

resource **socket_create_listen** (int port [, int backlog]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_create_pair (PHP 4 >= 4.1.0)

Crea una coppia di socket non distinguibili e li memorizza in fd.

bool **socket_create_pair** (int domain, int type, int protocol, array &fd) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_get_option (unknown)

Gets socket options for the socket

mixed **socket_get_option** (resource socket, int level, int optname) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

Nota: This function used to be called `socket_getopt()` prior to PHP 4.3.0

socket_getpeername (PHP 4 >= 4.1.0)

Dato un fd, memorizza la stringa rappresentante sa.sin_addr ed il valore di sa.sin_port in addr e port, descrivendo il lato remoto di un socket

bool **socket_getpeername** (resource socket, string &addr [, int &port]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_getsockname (PHP 4 >= 4.1.0)

Dato un fd, memorizza la stringa rappresentante sa.sin_addr ed il valore di sa.sin_port in addr e port, descrivendo il lato locale di un socket

bool **socket_getsockname** (resource socket, string &addr [, int &port]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_iovec_add (PHP 4 >= 4.1.0)

Aggiunge un nuovo vettore all'array ricevuti o inviati

bool **socket_iovec_add** (resource iovec, int iov_len) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_iovec_alloc (PHP 4 >= 4.1.0)

...]) Costruice una struttura 'struct iovec' da utilizzare con sendmsg, recvmsg, writev, e readv

resource **socket_iovec_alloc** (int num_vectors [, int]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_iovec_delete (PHP 4 >= 4.1.0)

Cancella un vettore da un array di vettori

bool **socket_iovec_delete** (resource iovec, int iov_pos) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_iovec_fetch (PHP 4 >= 4.1.0)

Restituisce i dati contenuti nella struttura iovec specificata da iovec_id[iovec_position]

string **socket_iovec_fetch** (resource iovec, int iovec_position) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_iovec_free (PHP 4 >= 4.1.0)

Libera la strutture iovec indicata da iovec_id

bool **socket_iovec_free** (resource iovec) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_iovec_set (PHP 4 >= 4.1.0)

Valorizza i dati contenuti in iovec_id[iovec_position] con new_val

bool **socket_iovec_set** (resource iovec, int iovec_position, string new_val) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_last_error (PHP 4 >= 4.1.0)

Restituisce/pulisce l'ultimo errore su un socket.

int **socket_last_error** (resource socket) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_listen (PHP 4 >= 4.1.0)

Attende una richiesta di connessione su un socket

int **socket_listen** (resource socket, int backlog) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Una volta creato il socket *socket* tramite la funzione socket_create(), ed eseguito il bind ad un nome con socket_bind(), lo si può mettere in ascolto di eventuali richieste di connessione. Tramite il parametro *backlog* si indica il numero massimo di connessioni in ingresso da tenere nella coda per l'elaborazione.

La funzione **socket_listen()** è disponibile solo per i socket di tipo `SOCK_STREAM` o `SOCK_SEQPACKET`.

La funzione restituisce zero se ha successo, oppure un codice di errore negativo in caso di errore. Questo codice può essere passato alla funzione socket_strerror() per ottenere una descrizione dell'errore.

Vedere anche socket_accept(), socket_bind(), socket_connect(), socket_create(), socket_get_status() e socket_strerror().

socket_read (PHP 4 >= 4.1.0)

Legge da un socket

string **socket_read** (resource socket_des, int length [, int type]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

La funzione **socket_read()** legge un numero di byte, indicati da *length*, da un socket *socket_des* creato dalla funzione socket_accept(). In alternativa si possono usare i caratteri \n, \t o \0 per indicare la fine della lettura. La funzione restituisce i dati oppure FALSE se **socket_read()** fallisce.

Il parametro opzionale *type* può assumere i seguenti valori:

- PHP_BINARY_READ - usa la funzione di sistema **socket_read()** (Default in PHP >= 4.1.0)
- PHP_NORMAL_READ - ferma la lettura in presenza di \n oppure \r. (Default in PHP <= 4.0.6)

Vedere anche socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_strerror(), socket_get_status() e socket_write().

socket_readv (PHP 4 >= 4.1.0)

Legge da un fd, utilizzando un array invia/ricevi definito da iovec_id

bool **socket_readv** (resource socket, resource iovec_id) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_recv (PHP 4 >= 4.1.0)

Riceve i dati da un socket collegato

string **socket_recv** (resource socket, int len, int flags) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l’uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_recvfrom (PHP 4 >= 4.1.0)

Riceve i dati da un socket, che sia connesso o meno

int **socket_recvfrom** (resource socket, string &buf, int len, int flags, string &name [, int &port]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_recvmsg (PHP 4 >= 4.1.0)

Funzione usata per ricevere messaggi da un socket, a prescindere che sia orientato alla connessione o meno

bool **socket_recvmsg** (resource socket, resource iovec, array &control, int &controllen, int &flags, string &addr [, int &port]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_select (PHP 4 >= 4.1.0)

Esegue la system call select() su un set con timeout indicato da tv_sec ed tv_usec

int **socket_select** (resource read_fd, resource write_fd, resource except_fd, int tv_sec [, int tv_usec]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_send (PHP 4 >= 4.1.0)

Invia i dati ad un socket collegato

int **socket_send** (resource socket, string buf, int len, int flags) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_sendmsg (PHP 4 >= 4.1.0)

Invia un messaggio ad un socket, a prescindere che sia orientato alla connessione o meno

bool **socket_sendmsg** (resource socket, resource iovec, int flags, string addr [, int port]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_sendto (PHP 4 >= 4.1.0)

Invia un messaggio ad un socket, a prescindere che sia connesso o meno

int **socket_sendto** (resource socket, string buf, int len, int flags, string addr [, int port]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_set_nonblock (PHP 4 >= 4.1.0)

Attiva la modalità "nonblocking" per il descrittore di file fd

bool **socket_set_nonblock** (resource socket) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_set_option (unknown)

Sets socket options for the socket

bool **socket_set_option** (resource socket, int level, int optname, int) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

Nota: This function used to be called `socket_setopt()` prior to PHP 4.3.0

socket_shutdown (PHP 4 >= 4.1.0)

Chiude un socket in ricezione, in invio o in entrambi i sensi.

bool **socket_shutdown** (resource socket [, int how]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

socket_strerror (PHP 4 >= 4.1.0)

Restituisce una stringa con la descrizione dell'errore.

string **socket_strerror** (int errno) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

La funzione **socket_strerror()** utilizza come proprio parametro *errno* il valore di ritorno di una funzione dei socket, e restituisce la corrispondente descrizione. E' utile potere spiegare cosa è accaduto quando qualcosa non funziona; ad esempio, invece di dovere cercare in tutto il sistema un file che contenga la spiegazione di '-111', si può semplicemente passare il valore a **socket_strerror()**, e ottenere la spiegazione di ciò che è accaduto.

Esempio 1. Esempio di uso di socket_strerror()

```
<?php
if (($socket = socket_create (AF_INET, SOCK_STREAM, 0)) < 0) {
   echo "socket_create() fallito: motivo: " . socket_strerror ($socket) . "\n";
}

if (($ret = socket_bind ($socket, '127.0.0.1', 80)) < 0) {
   echo "socket_bind() fallito: motivo: " . socket_strerror ($ret) . "\n";
}
?>
```

Dall'esempio precedente (se non viene eseguito con i privilegi di root) ci si aspetta il seguente messaggio:

```
socket_bind() fallito: motivo: Permission denied
```

Vedere anche socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_create() e socket_get_status().

socket_write (PHP 4 >= 4.1.0)

Scrive su un socket.

int **socket_write** (resource socket_des, string &buffer, int length) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

La funzione **socket_write()** scrive sul socket *`socket_des`* un numero di byte indicato da *`length`* tratti dal campo *`&buffer`*.

Vedere anche socket_accept(), socket_bind(), socket_connect(), socket_listen(), socket_read(), socket_strerror() e socket_get_status().

socket_writev (PHP 4 >= 4.1.0)

Scrive su un descrittore di file, fd, usando un array invia/ricevi definito da iovec_id

bool **socket_writev** (resource socket, resource iovec_id) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

XCVI. String functions

These functions all manipulate strings in various ways. Some more specialized sections can be found in the regular expression and URL handling sections.

For information on how strings behave, especially with regard to usage of single quotes, double quotes, and escape sequences, see the Strings entry in the Types section of the manual.

For even more powerful string handling and manipulating functions take a look at the POSIX regular expression functions and the Perl compatible regular expression functions

addcslashes (PHP 4 >= 4.0.0)

Quote string with slashes in a C style

string **addcslashes** (string str, string charlist) \linebreak

Returns a string with backslashes before characters that are listed in *charlist* parameter. It escapes `\n`, `\r` etc. in C-like style, characters with ASCII code lower than 32 and higher than 126 are converted to octal representation.

Be careful if you choose to escape characters 0, a, b, f, n, r, t and v. They will be converted to \0, \a, \b, \f, \n, \r, \t and \v. In PHP \0 (`NULL`), \r (carriage return), \n (newline) and \t (tab) are predefined escape sequences, while in C all of these are predefined escape sequences.

charlist like "\0..\37", which would escape all characters with ASCII code between 0 and 31.

Esempio 1. addcslashes() example

```
$escaped = addcslashes($not_escaped, "\0..\37!@\177..\377");
```

When you define a sequence of characters in the charlist argument make sure that you know what characters come between the characters that you set as the start and end of the range.

```
echo addcslashes('foo[ ]', 'A..z');
// output:  \f\o\o\[ \]
// All upper and lower-case letters will be escaped
// ... but so will the [\]^_‘ and any tabs, line
// feeds, carriage returns, etc.
```

Also, if the first character in a range has a lower ASCII value than the second character in the range, no range will be constructed. Only the start, end and period characters will be escaped. Use the ord() function to find the ASCII value for a character.

```
echo addcslashes("zoo['.']", 'z..A');
// output:  \zoo['\.']
```

See also stripcslashes(), stripslashes(), htmlspecialchars(), and quotemeta().

addslashes (PHP 3, PHP 4 >= 4.0.0)

Quote string with slashes

string **addslashes** (string str) \linebreak

Returns a string with backslashes before characters that need to be quoted in database queries etc. These characters are single quote (`'`), double quote (`"`), backslash (`\`) and NUL (the `NULL` byte).

> **Nota:** magic_quotes_gpc is ON by default.

See also stripslashes(), htmlspecialchars(), and quotemeta().

bin2hex (PHP 3>= 3.0.9, PHP 4 >= 4.0.0)

Convert binary data into hexadecimal representation

string **bin2hex** (string str) \linebreak

Returns an ASCII string containing the hexadecimal representation of *`str`*. The conversion is done byte-wise with the high-nibble first.

See also pack() and unpack().

chop (PHP 3, PHP 4 >= 4.0.0)

Alias of rtrim()

This function is an alias of rtrim().

> **Nota: chop()** is different than the Perl `chop()` function, which removes the last character in the string.

chr (PHP 3, PHP 4 >= 4.0.0)

Return a specific character

string **chr** (int ascii) \linebreak

Returns a one-character string containing the character specified by *`ascii`*.

Esempio 1. chr() example

```
$str .= chr(27); /* add an escape character at the end of $str */

/* Often this is more useful */

$str = sprintf("The string ends in escape: %c", 27);
```

You can find an ASCII-table over here:
http://www.mindspring.com/~jc1/serial/Resources/ASCII.html.

This function complements ord(). See also sprintf() with a format string of `%c`.

chunk_split (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Split a string into smaller chunks

string **chunk_split** (string body [, int chunklen [, string end]]) \linebreak

Can be used to split a string into smaller chunks which is useful for e.g. converting base64_encode output to match RFC 2045 semantics. It inserts *`end`* (defaults to "\r\n") every *`chunklen`* characters (defaults to 76). It returns the new string leaving the original string untouched.

Esempio 1. chunk_split() example

```
# format $data using RFC 2045 semantics

$new_string = chunk_split(base64_encode($data));
```

See also explode(), split() and wordwrap().

convert_cyr_string (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Convert from one Cyrillic character set to another

string **convert_cyr_string** (string str, string from, string to) \linebreak

This function returns the given string converted from one Cyrillic character set to another. The *`from`* and *`to`* arguments are single characters that represent the source and target Cyrillic character sets. The supported types are:

- k - koi8-r
- w - windows-1251

- i - iso8859-5
- a - x-cp866
- d - x-cp866
- m - x-mac-cyrillic

count_chars (PHP 4 >= 4.0.0)

Return information about characters used in a string

mixed **count_chars** (string string [, int mode]) \linebreak

Counts the number of occurrences of every byte-value (0..255) in *string* and returns it in various ways. The optional parameter *Mode* default to 0. Depending on *mode* **count_chars()** returns one of the following:

- 0 - an array with the byte-value as key and the frequency of every byte as value.
- 1 - same as 0 but only byte-values with a frequency greater than zero are listed.
- 2 - same as 0 but only byte-values with a frequency equal to zero are listed.
- 3 - a string containing all used byte-values is returned.
- 4 - a string containing all not used byte-values is returned.

crc32 (PHP 4)

Calculates the crc32 polynomial of a string

int **crc32** (string str) \linebreak

Generates the cyclic redundancy checksum polynomial of 32-bit lengths of the *str*. This is usually used to validate the integrity of data being transmitted.

See also: md5()

crypt (PHP 3, PHP 4 >= 4.0.0)

One-way string encryption (hashing)

string **crypt** (string str [, string salt]) \linebreak

crypt() will return an encrypted string using the standard Unix DES-based encryption algorithm or alternative algorithms that may be available on the system. Arguments are a string to be encrypted and an optional salt string to base the encryption on. See the Unix man page for your crypt function for more information.

If the salt argument is not provided, one will be randomly generated by PHP.

Some operating systems support more than one type of encryption. In fact, sometimes the standard DES-based encryption is replaced by an MD5-based encryption algorithm. The encryption type is triggered by the salt argument. At install time, PHP determines the capabilities of the crypt function and will accept salts for other encryption types. If no salt is provided, PHP will auto-generate a standard two character salt by default, unless the default encryption type on the system is MD5, in which case a random MD5-compatible salt is generated. PHP sets a constant named CRYPT_SALT_LENGTH which tells you whether a regular two character salt applies to your system or the longer twelve character salt is applicable.

If you are using the supplied salt, you should be aware that the salt is generated once. If you are calling this function recursively, this may impact both appearance and security.

The standard DES-based encryption **crypt()** returns the salt as the first two characters of the output. It also only uses the first eight characters of *str*, so longer strings that start with the same eight characters will generate the same result (when the same salt is used).

On systems where the crypt() function supports multiple encryption types, the following constants are set to 0 or 1 depending on whether the given type is available:

- CRYPT_STD_DES - Standard DES-based encryption with a two character salt
- CRYPT_EXT_DES - Extended DES-based encryption with a nine character salt
- CRYPT_MD5 - MD5 encryption with a twelve character salt starting with 1
- CRYPT_BLOWFISH - Blowfish encryption with a sixteen character salt starting with 2

Nota: There is no decrypt function, since **crypt()** uses a one-way algorithm.

Esempio 1. crypt() examples

```
<?php
$password = crypt("My1sTpassword"); # let salt be generated

# You should pass the entire results of crypt() as the salt for comparing a
# password, to avoid problems when different hashing algorithms are used. (As
# it says above, standard DES-based password hashing uses a 2-character salt,
# but MD5-based hashing uses 12.)
if (crypt($user_input,$password) == $password) {
   echo "Password verified!";
}
?>
```

See also md5() and the Mcrypt extension.

echo (unknown)

Output one or more strings

echo (string arg1 [, string argn...]) \linebreak

Outputs all parameters.

echo() is not actually a function (it is a language construct) so you are not required to use parentheses with it. In fact, if you want to pass more than one parameter to echo, you must not enclose the parameters within parentheses. It is not possible to use **echo()** in a variable function context, but you can use print() instead.

Esempio 1. echo() examples

```
<?php
echo "Hello World";

echo "This spans
multiple lines. The newlines will be
output as well";

echo "This spans\nmultiple lines. The newlines will be\noutput as well.";

echo "escaping characters is done \"Like this\"."

//You can use variables inside of an echo statement
$foo = "foobar";
$bar = "barbaz";

echo "foo is $foo"; // foo is foobar

// Using single quotes will print the variable name, not the value
echo 'foo is $foo'; // foo is $foo

// If you are not using any other characters, you can just echo variables
echo $foo;          // foobar
echo $foo,$bar;     // foobarbarbaz

echo <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;

// because echo is not a function, following code is invalid.
($some_var) ? echo('true'): echo('false');

// However, the following examples will work:
($some_var) ? print('true'): print('false'); // print is a function
echo $some_var ? 'true': 'false'; // changing the statement around
?>
```

echo() also has a shortcut syntax, where you can immediately follow the opening tag with an equals sign.

```
I have <?=$foo?> foo.
```

Nota: This short syntax only works with the short_open_tag configuration setting enabled.

See also: print(), printf(), and flush().

explode (PHP 3, PHP 4 >= 4.0.0)

Split a string by string

array **explode** (string separator, string string [, int limit]) \linebreak

Returns an array of strings, each of which is a substring of *string* formed by splitting it on boundaries formed by the string *separator*. If *limit* is set, the returned array will contain a maximum of *limit* elements with the last element containing the rest of *string*.

If *separator* is an empty string (""), **explode()** will return FALSE. If *separator* contains a value that is not contained in *string*, then **explode()** will return an array containing *string*.

Nota: The *limit* parameter was added in PHP 4.0.1

Esempio 1. explode() examples

```
$pizza = "piece1 piece2 piece3 piece4 piece5 piece6";
$pieces = explode(" ", $pizza);

$data = "foo:*:1023:1000::/home/foo:/bin/sh";
list($user,$pass,$uid,$gid,$gecos,$home,$shell) = explode(":",$data);
```

Nota: Although implode() can for historical reasons accept its parameters in either order, **explode()** cannot. You must ensure that the *separator* argument comes before the *string* argument.

See also preg_split(), spliti(), split(), and implode().

get_html_translation_table (PHP 4 >= 4.0.0)

Returns the translation table used by htmlspecialchars() and htmlentities()

string **get_html_translation_table** (int table [, int quote_style]) \linebreak

get_html_translation_table() will return the translation table that is used internally for htmlspecialchars() and htmlentities(). There are two new defines (*HTML_ENTITIES*, *HTML_SPECIALCHARS*) that allow you to specify the table you want. And as in the htmlspecialchars() and htmlentities() functions you can optionally specify the quote_style you are working with. The default is ENT_COMPAT mode. See the description of these modes in htmlspecialchars().

Esempio 1. Translation Table Example

```
$trans = get_html_translation_table(HTML_ENTITIES);
$str = "Hallo & <Frau> & Krämer";
$encoded = strtr($str, $trans);
```

The $encoded variable will now contain: "Hallo & <Frau> & Krämer".

The cool thing is using array_flip() to change the direction of the translation.

```
$trans = array_flip($trans);
$original = strtr($encoded, $trans);
```

The content of $original would be: "Hallo & <Frau> & Krämer".

See also: htmlspecialchars(), htmlentities(), strtr(), and array_flip().

get_meta_tags (PHP 3>= 3.0.4, PHP 4 >= 4.0.0)

Extracts all meta tag content attributes from a file and returns an array

array **get_meta_tags** (string filename [, int use_include_path]) \linebreak

Opens *filename* and parses it line by line for <meta> tags of the form

Esempio 1. Meta Tags Example

```
<meta name="author" content="name">
<meta name="tags" content="php3 documentation">
</head> <!-- parsing stops here -->
```

(pay attention to line endings - PHP uses a native function to parse the input, so a Mac file won't work on Unix).

The value of the name property becomes the key, the value of the content property becomes the value of the returned array, so you can easily use standard array functions to traverse it or access single values. Special characters in the value of the name property are substituted with '_', the rest is converted to lower case.

Setting *`use_include_path`* to 1 will result in PHP trying to open the file along the standard include path.

hebrev (PHP 3, PHP 4 >= 4.0.0)

Convert logical Hebrew text to visual text

string **hebrev** (string hebrew_text [, int max_chars_per_line]) \linebreak

The optional parameter *`max_chars_per_line`* indicates maximum number of characters per line will be output. The function tries to avoid breaking words.

See also hebrevc()

hebrevc (PHP 3, PHP 4 >= 4.0.0)

Convert logical Hebrew text to visual text with newline conversion

string **hebrevc** (string hebrew_text [, int max_chars_per_line]) \linebreak

This function is similar to hebrev() with the difference that it converts newlines (\n) to "
\n". The optional parameter *`max_chars_per_line`* indicates maximum number of characters per line will be output. The function tries to avoid breaking words.

See also hebrev()

htmlentities (PHP 3, PHP 4 >= 4.0.0)

Convert all applicable characters to HTML entities

string **htmlentities** (string string [, int quote_style [, string charset]]) \linebreak

This function is identical to htmlspecialchars() in all ways, except that all characters which have HTML character entity equivalents are translated into these entities. Like htmlspecialchars(), it takes an optional second argument which indicates what should be done with single and double quotes. `ENT_COMPAT` (the default) will only convert double-quotes and leave single-quotes alone. `ENT_QUOTES` will convert both double and single quotes, and `ENT_NOQUOTES` will leave both double and single quotes unconverted.

At present, the ISO-8859-1 character set is used as default. Support for the optional second argument was added in PHP 3.0.17 and PHP 4.0.3.

Like htmlspecialchars(), it takes an optional third argument which defines character set used in conversion. Support for this argument was added in PHP 4.1.0.

There is no reverse of this function. However, you can create one on your own. Here is an example of how to do this.

Esempio 1. Reverse of htmlentities()

```
<?php
function unhtmlentities ($string)
{
$trans_tbl = get_html_translation_table (HTML_ENTITIES);
$trans_tbl = array_flip ($trans_tbl);
return strtr ($string, $trans_tbl);
}
?>
```

See also htmlspecialchars() and nl2br().

htmlspecialchars (PHP 3, PHP 4 >= 4.0.0)

Convert special characters to HTML entities

string **htmlspecialchars** (string string [, int quote_style [, string charset]]) \linebreak

Certain characters have special significance in HTML, and should be represented by HTML entities if they are to preserve their meanings. This function returns a string with some of these conversions made; the translations made are those most useful for everyday web programming. If you require all HTML character entities to be translated, use htmlentities() instead.

This function is useful in preventing user-supplied text from containing HTML markup, such as in a message board or guest book application. The optional second argument, quote_style, tells the function what to do with single and double quote characters. The default mode, ENT_COMPAT, is the backwards compatible mode which only translates the double-quote character and leaves the single-quote untranslated. If ENT_QUOTES is set, both single and double quotes are translated and if ENT_NOQUOTES is set neither single nor double quotes are translated.

The translations performed are:

- ’&’ (ampersand) becomes ’&’
- ’"’ (double quote) becomes ’"’ when ENT_NOQUOTES is not set.
- ”’ (single quote) becomes ’'’ only when ENT_QUOTES is set.
- ’<’ (less than) becomes ’<’
- ’>’ (greater than) becomes ’>’

Esempio 1. htmlspecialchars() example

```
$new = htmlspecialchars("<a href='test'>Test</a>", ENT_QUOTES);
```

Note that this function does not translate anything beyond what is listed above. For full entity translation, see htmlentities(). Support for the optional second argument was added in PHP 3.0.17 and PHP 4.0.3.

The third argument defines character set used in conversion. The default character set is ISO-8859-1. Support for this third argument was added in PHP 4.1.0.

See also htmlentities() and nl2br().

implode (PHP 3, PHP 4 >= 4.0.0)

Join array elements with a string

string **implode** (string glue, array pieces) \linebreak

Returns a string containing a string representation of all the array elements in the same order, with the glue string between each element.

Esempio 1. implode() example

```
$colon_separated = implode(":", $array);
```

> **Nota: implode()** can, for historical reasons, accept its parameters in either order. For consistency with explode(), however, it may be less confusing to use the documented order of arguments.

See also explode(), join(), and split().

join (PHP 3, PHP 4 >= 4.0.0)

Join array elements with a string

string **join** (string glue, array pieces) \linebreak

join() is an alias to implode(), and is identical in every way.

See also explode(), implode(), and split().

levenshtein (PHP 3>= 3.0.17, PHP 4)

Calculate Levenshtein distance between two strings

int **levenshtein** (string str1, string str2) \linebreak int **levenshtein** (string str1, string str2, int cost_ins, int cost_rep, int cost_del) \linebreak int **levenshtein** (string str1, string str2, function cost) \linebreak

This function returns the Levenshtein-Distance between the two argument strings or -1, if one of the argument strings is longer than the limit of 255 characters (255 should be more than enough for name or dictionary comparison, and nobody serious would be doing genetic analysis with PHP).

The Levenshtein distance is defined as the minimal number of characters you have to replace, insert or delete to transform *`str1`* into *`str2`*. The complexity of the algorithm is `O(m*n)`, where `n` and `m` are the length of *`str1`* and *`str2`* (rather good when compared to similar_text(), which is O(max(n,m)**3), but still expensive).

In its simplest form the function will take only the two strings as parameter and will calculate just the number of insert, replace and delete operations needed to transform *`str1`* into *`str2`*.

A second variant will take three additional parameters that define the cost of insert, replace and delete operations. This is more general and adaptive than variant one, but not as efficient.

The third variant (which is not implemented yet) will be the most general and adaptive, but also the slowest alternative. It will call a user-supplied function that will determine the cost for every possible operation.

The user-supplied function will be called with the following arguments:

- operation to apply: ’I’, ’R’ or ’D’
- actual character in string 1
- actual character in string 2
- position in string 1
- position in string 2
- remaining characters in string 1
- remaining characters in string 2

The user-supplied function has to return a positive integer describing the cost for this particular operation, but it may decide to use only some of the supplied arguments.

The user-supplied function approach offers the possibility to take into account the relevance of and/or difference between certain symbols (characters) or even the context those symbols appear in to determine the cost of insert, replace and delete operations, but at the cost of losing all optimizations done regarding cpu register utilization and cache misses that have been worked into the other two variants.

See also soundex(), similar_text(), and metaphone().

localeconv (PHP 4 >= 4.0.5)

Get numeric formatting information

array **localeconv** (void) \linebreak

Returns an associative array containing localized numeric and monetary formatting information.

localeconv() returns data based upon the current locale as set by setlocale(). The associative array that is returned contains the following fields:

Array element	**Description**
decimal_point	Decimal point character
thousands_sep	Thousands separator
grouping	Array containing numeric groupings
int_curr_symbol	International currency symbol (i.e. USD)
currency_symbol	Local currency symbol (i.e. $)
mon_decimal_point	Monetary decimal point character
mon_thousands_sep	Monetary thousands separator
mon_grouping	Array containing monetary groupings
positive_sign	Sign for positive values
negative_sign	Sign for negative values
int_frac_digits	International fractional digits
frac_digits	Local fractional digits
p_cs_precedes	`TRUE` if currency_symbol precedes a positive value, `FALSE` if it succeeds one
p_sep_by_space	`TRUE` if a space separates currency_symbol from a positive value, `FALSE` otherwise
n_cs_precedes	`TRUE` if currency_symbol precedes a negative value, `FALSE` if it succeeds one
n_sep_by_space	`TRUE` if a space separates currency_symbol from a negative value, `FALSE` otherwise
p_sign_posn	0 Parentheses surround the quantity and currency_symbol 1 The sign string precedes the quantity and currency_symbol 2 The sign string succeeds the quantity and currency_symbol 3 The sign string immediately precedes the currency_symbol 4 The sign string immediately succeeds the currency_symbol

Array element	Description
n_sign_posn	0 Parentheses surround the quantity and currency_symbol 1 The sign string precedes the quantity and currency_symbol 2 The sign string succeeds the quantity and currency_symbol 3 The sign string immediately precedes the currency_symbol 4 The sign string immediately succeeds the currency_symbol

The grouping fields contain arrays that define the way numbers should be grouped. For example, the grouping field for the en_US locale, would contain a 2 item array with the values 3 and 3. The higher the index in the array, the farther left the grouping is. If an array element is equal to CHAR_MAX, no further grouping is done. If an array element is equal to 0, the previous element should be used.

Esempio 1. localeconv() example

```
setlocale(LC_ALL, "en_US");

$locale_info = localeconv();

echo "<PRE>\n";
echo "--------------------------------------------\n";
echo "  Monetary information for current locale:  \n";
echo "--------------------------------------------\n\n";

echo "int_curr_symbol:   {$locale_info["int_curr_symbol"]}\n";
echo "currency_symbol:   {$locale_info["currency_symbol"]}\n";
echo "mon_decimal_point: {$locale_info["mon_decimal_point"]}\n";
echo "mon_thousands_sep: {$locale_info["mon_thousands_sep"]}\n";
echo "positive_sign:     {$locale_info["positive_sign"]}\n";
echo "negative_sign:     {$locale_info["negative_sign"]}\n";
echo "int_frac_digits:   {$locale_info["int_frac_digits"]}\n";
echo "frac_digits:       {$locale_info["frac_digits"]}\n";
echo "p_cs_precedes:     {$locale_info["p_cs_precedes"]}\n";
echo "p_sep_by_space:    {$locale_info["p_sep_by_space"]}\n";
echo "n_cs_precedes:     {$locale_info["n_cs_precedes"]}\n";
echo "n_sep_by_space:    {$locale_info["n_sep_by_space"]}\n";
echo "p_sign_posn:       {$locale_info["p_sign_posn"]}\n";
echo "n_sign_posn:       {$locale_info["n_sign_posn"]}\n";
echo "</PRE>\n";
```

The constant CHAR_MAX is also defined for the use mentioned above.

See also: setlocale().

ltrim (PHP 3, PHP 4 >= 4.0.0)

Strip whitespace from the beginning of a string

string **ltrim** (string str [, string charlist]) \linebreak

Nota: The second parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the beginning of *str*. Without the second parameter, **ltrim()** will strip these characters:

- " " (ASCII 32 (0x20)), an ordinary space.
- "\t" (ASCII 9 (0x09)), a tab.
- "\n" (ASCII 10 (0x0A)), a new line (line feed).
- "\r" (ASCII 13 (0x0D)), a carriage return.
- "\0" (ASCII 0 (0x00)), the NUL-byte.
- "\x0B" (ASCII 11 (0x0B)), a vertical tab.

You can also specify the characters you want to strip, by means of the *charlist* parameter. Simply list all characters that you want to be stripped. With `..` you can specify a range of characters.

Esempio 1. Usage example of ltrim()

```
<?php

$text = "\t\tThese are a few words :) ...  ";
$trimmed = ltrim($text);
// $trimmed = "These are a few words :) ...  "
$trimmed = ltrim($text," \t.");
// $trimmed = "These are a few words :) ...  "
$clean = ltrim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the beginning of $binary
// (from 0 to 31 inclusive)

?>
```

See also trim() and rtrim().

md5 (PHP 3, PHP 4 >= 4.0.0)

Calculate the md5 hash of a string

string **md5** (string str) \linebreak

Calculates the MD5 hash of *str* using the RSA Data Security, Inc. MD5 Message-Digest Algorithm (http://www.faqs.org/rfcs/rfc1321.html), and returns that hash. The hash is a 32-character hexadecimal number.

See also: crc32() and md5_file()

md5_file (PHP 4 >= 4.2.0)

Calculates the md5 hash of a given filename

string **md5_file** (string filename) \linebreak

Calculates the MD5 hash of the specified *filename* using the RSA Data Security, Inc. MD5 Message-Digest Algorithm (http://www.faqs.org/rfcs/rfc1321.html), and returns that hash.

This function has the same purpose of the command line utility md5sum.

See also: md5() and crc32()

metaphone (PHP 4 >= 4.0.0)

Calculate the metaphone key of a string

string **metaphone** (string str) \linebreak

Calculates the metaphone key of *str*.

Similar to soundex() metaphone creates the same key for similar sounding words. It's more accurate than soundex() as it knows the basic rules of English pronunciation. The metaphone generated keys are of variable length.

Metaphone was developed by Lawrence Philips <lphilips@verity.com>. It is described in ["Practical Algorithms for Programmers", Binstock & Rex, Addison Wesley, 1995].

nl_langinfo (PHP 4 >= 4.1.0)

Query language and locale information

string **nl_langinfo** (int item) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

nl2br (PHP 3, PHP 4 >= 4.0.0)

Inserts HTML line breaks before all newlines in a string

string **nl2br** (string string) \linebreak

Returns *string* with '<br />' inserted before all newlines.

> **Nota:** Starting with PHP 4.0.5, **nl2br()** is now XHTML compliant. All versions before 4.0.5 will return *string* with '<br>' inserted before newlines instead of '<br />'.

See also htmlspecialchars(), htmlentities() and wordwrap().

ord (PHP 3, PHP 4 >= 4.0.0)

Return ASCII value of character

int **ord** (string string) \linebreak

Returns the ASCII value of the first character of *string*. This function complements chr().

Esempio 1. ord() example

```
if (ord($str) == 10) {
    echo "The first character of \$str is a line feed.\n";
}
```

You can find an ASCII-table over here:
http://www.mindspring.com/~jc1/serial/Resources/ASCII.html.

See also chr().

parse_str (PHP 3, PHP 4 >= 4.0.0)

Parses the string into variables

void **parse_str** (string str [, array arr]) \linebreak

Parses *str* as if it were the query string passed via an URL and sets variables in the current scope. If the second parameter *arr* is present, variables are stored in this variable as an array elements instead.

Nota: Support for the optional second parameter was added in PHP 4.0.3.

Esempio 1. Using parse_str()

```
$str = "first=value&second[]=this+works&second[]=another";
parse_str($str);
echo $first;     /* prints "value" */
echo $second[0]; /* prints "this works" */
echo $second[1]; /* prints "another" */
```

See also set_magic_quotes_runtime() and urldecode().

print (unknown)

Output a string

print (string arg) \linebreak

Outputs *arg*. Restituisce TRUE in caso di successo, FALSE in caso di fallimento.

print() is not actually a real function (it is a language construct) so you are not required to use parentheses with it. But **print()**, opposed to echo(), can be called using a variable function.

Esempio 1. print() examples

```
<?php
print("Hello World");

print "print() also works without parentheses.";

print "This spans
multiple lines. The newlines will be
output as well";

print "This spans\nmultiple lines. The newlines will be\noutput as well.";

print "escaping characters is done \"Like this\".";

// You can use variables inside of an print statement
$foo = "foobar";
$bar = "barbaz";

print "foo is $foo"; // foo is foobar

// Using single quotes will print the variable name, not the value
print ’foo is $foo’; // foo is $foo
```

```
// If you are not using any other characters, you can just print variables
print $foo;          // foobar

print <<<END
This uses the "here document" syntax to output
multiple lines with $variable interpolation. Note
that the here document terminator must appear on a
line with just a semicolon no extra whitespace!
END;
?>
```

See also echo(), printf(), and flush().

printf (PHP 3, PHP 4 >= 4.0.0)

Output a formatted string

void **printf** (string format [, mixed args]) \linebreak

Produces output according to *format*, which is described in the documentation for sprintf().

See also: print(), sprintf(), sscanf(), fscanf(), and flush().

quoted_printable_decode (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Convert a quoted-printable string to an 8 bit string

string **quoted_printable_decode** (string str) \linebreak

This function returns an 8-bit binary string corresponding to the decoded quoted printable string. This function is similar to imap_qprint(), except this one does not require the IMAP module to work.

quotemeta (PHP 3, PHP 4 >= 4.0.0)

Quote meta characters

string **quotemeta** (string str) \linebreak

Returns a version of str with a backslash character (\) before every character that is among these:

```
. \\ + * ? [ ^ ] ( $ )
```

See also addslashes(), htmlentities(), htmlspecialchars(), nl2br(), and stripslashes().

rtrim (PHP 3, PHP 4 >= 4.0.0)

Strip whitespace from the end of a string

string **rtrim** (string str [, string charlist]) \linebreak

> **Nota:** The second parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the end of *str*. Without the second parameter, **rtrim()** will strip these characters:

- " " (ASCII 32 (0x20)), an ordinary space.
- "\t" (ASCII 9 (0x09)), a tab.
- "\n" (ASCII 10 (0x0A)), a new line (line feed).
- "\r" (ASCII 13 (0x0D)), a carriage return.
- "\0" (ASCII 0 (0x00)), the NUL-byte.
- "\x0B" (ASCII 11 (0x0B)), a vertical tab.

You can also specify the characters you want to strip, by means of the *charlist* parameter. Simply list all characters that you want to be stripped. With `..` you can specify a range of characters.

Esempio 1. Usage example of rtrim()

```
<?php

$text = "\t\tThese are a few words :) ...  ";
$trimmed = rtrim($text);
// $trimmed = "\t\tThese are a few words :) ..."
$trimmed = rtrim($text," \t.");
// $trimmed = "\t\tThese are a few words :)"
$clean = rtrim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the end of $binary
// (from 0 to 31 inclusive)

?>
```

See also trim() and ltrim().

setlocale (PHP 3, PHP 4 >= 4.0.0)

Set locale information

string **setlocale** (mixed category, string locale) \linebreak

Category is a named constant (or string) specifying the category of the functions affected by the locale setting:

- LC_ALL for all of the below
- LC_COLLATE for string comparison, see strcoll()
- LC_CTYPE for character classification and conversion, for example strtoupper()
- LC_MONETARY for localeconv()
- LC_NUMERIC for decimal separator (See also: localeconv())
- LC_TIME for date and time formatting with strftime()

If *locale* is the empty string `""`, the locale names will be set from the values of environment variables with the same names as the above categories, or from "LANG".

If locale is zero or `"0"`, the locale setting is not affected, only the current setting is returned.

Setlocale returns the new current locale, or `FALSE` if the locale functionality is not implemented in the platform, the specified locale does not exist or the category name is invalid. An invalid category name also causes a warning message.

Esempio 1. setlocale() Examples

```
<?php
    /* Set locale to Dutch */
    setlocale (LC_ALL, ’nl_NL’);

    /* Output: vrijdag 22 december 1978 */
    echo strftime ("%A %e %B %Y", mktime (0, 0, 0, 12, 22, 1978));
?>
```

similar_text (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Calculate the similarity between two strings

int **similar_text** (string first, string second [, float percent]) \linebreak

This calculates the similarity between two strings as described in Oliver [1993]. Note that this implementation does not use a stack as in Oliver’s pseudo code, but recursive calls which may or may not speed up the whole process. Note also that the complexity of this algorithm is O(N**3) where N is the length of the longest string.

By passing a reference as third argument, **similar_text()** will calculate the similarity in percent for you. It returns the number of matching chars in both strings.

soundex (PHP 3, PHP 4 >= 4.0.0)

Calculate the soundex key of a string

string **soundex** (string str) \linebreak

Calculates the soundex key of *str*.

Soundex keys have the property that words pronounced similarly produce the same soundex key, and can thus be used to simplify searches in databases where you know the pronunciation but not the spelling. This soundex function returns a string 4 characters long, starting with a letter.

This particular soundex function is one described by Donald Knuth in "The Art Of Computer Programming, vol. 3: Sorting And Searching", Addison-Wesley (1973), pp. 391-392.

Esempio 1. Soundex Examples

```
soundex("Euler") == soundex("Ellery") == 'E460';
soundex("Gauss") == soundex("Ghosh") == 'G200';
soundex("Hilbert") == soundex("Heilbronn") == 'H416';
soundex("Knuth") == soundex("Kant") == 'K530';
soundex("Lloyd") == soundex("Ladd") == 'L300';
soundex("Lukasiewicz") == soundex("Lissajous") == 'L222';
```

See also levenshtein(), metaphone(), and similar_text().

sprintf (PHP 3, PHP 4 >= 4.0.0)

Return a formatted string

string **sprintf** (string format [, mixed args]) \linebreak

Returns a string produced according to the formatting string *format*.

The format string is composed of zero or more directives: ordinary characters (excluding `%`) that are copied directly to the result, and *conversion specifications*, each of which results in fetching its own parameter. This applies to both **sprintf()** and printf().

Each conversion specification consists of a percent sign (`%`), followed by one or more of these elements, in order:

1. An optional *padding specifier* that says what character will be used for padding the results to the right string size. This may be a space character or a `0` (zero character). The default is to pad

with spaces. An alternate padding character can be specified by prefixing it with a single quote (`'`). See the examples below.

2. An optional *alignment specifier* that says if the result should be left-justified or right-justified. The default is right-justified; a `-` character here will make it left-justified.
3. An optional number, a *width specifier* that says how many characters (minimum) this conversion should result in.
4. An optional *precision specifier* that says how many decimal digits should be displayed for floating-point numbers. This option has no effect for other types than float. (Another function useful for formatting numbers is number_format().)
5. A *type specifier* that says what type the argument data should be treated as. Possible types:
   `%` - a literal percent character. No argument is required.
   `b` - the argument is treated as an integer, and presented as a binary number.
   `c` - the argument is treated as an integer, and presented as the character with that ASCII value.
   `d` - the argument is treated as an integer, and presented as a (signed) decimal number.
   `u` - the argument is treated as an integer, and presented as an unsigned decimal number.
   `f` - the argument is treated as a float, and presented as a floating-point number.
   `o` - the argument is treated as an integer, and presented as an octal number.
   `s` - the argument is treated as and presented as a string.
   `x` - the argument is treated as an integer and presented as a hexadecimal number (with lowercase letters).
   `X` - the argument is treated as an integer and presented as a hexadecimal number (with uppercase letters).

As of PHP version 4.0.6 the format string supports argument numbering/swapping. Here is an example:

Esempio 1. Argument swapping

```
$format = "There are %d monkeys in the %s";
printf($format,$num,$location);
```

This might output, "There are 5 monkeys in the tree". But imagine we are creating a format string in a separate file, commonly because we would like to internationalize it and we rewrite it as:

Esempio 2. Argument swapping

```
$format = "The %s contains %d monkeys";
printf($format,$num,$location);
```

We now have a problem. The order of the placeholders in the format string does not match the order of the arguments in the code. We would like to leave the code as is and simply indicate in the format string which arguments the placeholders refer to. We would write the format string like this instead:

Esempio 3. Argument swapping

```
$format = "The %2\$s contains %1\$d monkeys";
printf($format,$num,$location);
```

An added benefit here is that you can repeat the placeholders without adding more arguments in the code. For example:

Esempio 4. Argument swapping

```
$format = "The %2\$s contains %1\$d monkeys.
           That’s a nice %2\$s full of %1\$d monkeys.";
printf($format, $num, $location);
```

See also: printf(), sscanf(), fscanf(), and number_format().

Esempio 5. sprintf(): zero-padded integers

```
$isodate = sprintf("%04d-%02d-%02d", $year, $month, $day);
```

Esempio 6. sprintf(): formatting currency

```
$money1 = 68.75;
$money2 = 54.35;
$money = $money1 + $money2;
// echo $money will output "123.1";
$formatted = sprintf("%01.2f", $money);
// echo $formatted will output "123.10"
```

sscanf (PHP 4)

Parses input from a string according to a format

mixed **sscanf** (string str, string format [, string var1]) \linebreak

The function **sscanf()** is the input analog of printf(). **sscanf()** reads from the string *`str`* and interprets it according to the specified *`format`*. If only two parameters were passed to this function, the values parsed will be returned as an array.

Esempio 1. sscanf() Example

```
// getting the serial number
$serial = sscanf("SN/2350001","SN/%d");
// and the date of manufacturing
$mandate = "January 01 2000";
list($month, $day, $year) = sscanf($mandate,"%s %d %d");
echo "Item $serial was manufactured on: $year-".substr($month,0,3)."-$day\n";
```

If optional parameters are passed, the function will return the number of assigned values. The optional parameters must be passed by reference.

Esempio 2. sscanf() - using optional parameters

```
// get author info and generate DocBook entry
$auth = "24\tLewis Carroll";
$n = sscanf($auth,"%d\t%s %s", &$id, &$first, &$last);
echo "<author id='$id'>
    <firstname>$first</firstname>
    <surname>$last</surname>
</author>\n";
```

See also: fscanf(), printf(), and sprintf().

str_pad (PHP 4)

Pad a string to a certain length with another string

string **str_pad** (string input, int pad_length [, string pad_string [, int pad_type]]) \linebreak

This functions returns the *input* string padded on the left, the right, or both sides to the specified padding length. If the optional argument *pad_string* is not supplied, the *input* is padded with spaces, otherwise it is padded with characters from *pad_string* up to the limit.

Optional argument *pad_type* can be STR_PAD_RIGHT, STR_PAD_LEFT, or STR_PAD_BOTH. If *pad_type* is not specified it is assumed to be STR_PAD_RIGHT.

If the value of *pad_length* is negative or less than the length of the input string, no padding takes place.

Esempio 1. str_pad() example

```
$input = "Alien";
print str_pad($input, 10);                      // produces "Alien     "
print str_pad($input, 10, "-=", STR_PAD_LEFT);  // produces "-=-=-Alien"
```

```
print str_pad($input, 10, "_", STR_PAD_BOTH);   // produces "__Alien___"
```

str_repeat (PHP 4 >= 4.0.0)

Repeat a string

string **str_repeat** (string input, int multiplier) \linebreak

Returns *input_str* repeated *multiplier* times. *multiplier* has to be greater than 0.

Esempio 1. str_repeat() example

```
echo str_repeat("-=", 10);
```

This will output "-=-=-=-=-=-=-=-=-=-=".

See also: for, str_pad(), and substr_count().

str_replace (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Replace all occurrences of the search string with the replacement string

mixed **str_replace** (mixed search, mixed replace, mixed subject) \linebreak

This function returns a string or an array with all occurences of *search* in *subject* replaced with the given *replace* value. If you don't need fancy replacing rules, you should always use this function instead of ereg_replace() or preg_replace().

In PHP 4.0.5 and later, every parameter to **str_replace()** can be an array.

If *subject* is an array, then the search and replace is performed with every entry of *subject*, and the return value is an array as well.

If *search* and *replace* are arrays, then **str_replace()** takes a value from each array and uses them to do search and replace on *subject*. If *replace* has fewer values than *search*, then an empty string is used for the rest of replacement values. If *search* is an array and *replace* is a string; then this replacement string is used for every value of *search*.

Esempio 1. str_replace() example

```
$bodytag = str_replace("%body%", "black", "<body text=%body%>");
```

This function is binary safe.

> **Nota: str_replace()** was added in PHP 3.0.6, but was buggy up until PHP 3.0.8.

See also ereg_replace(), preg_replace(), and strtr().

str_rot13 (PHP 4 >= 4.2.0)

Perform the rot13 transform on a string

string **str_rot13** (string str) \linebreak

This function performs the ROT13 encoding on the *str* argument and returns the resulting string. The ROT13 encoding simply shifts every letter by 13 places in the alphabet while leaving non-alpha characters untouched. Encoding and decoding are done by the same function, passing an encoded string as argument will return the original version.

strcasecmp (PHP 3>= 3.0.2, PHP 4 >= 4.0.0)

Binary safe case-insensitive string comparison

int **strcasecmp** (string str1, string str2) \linebreak

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Esempio 1. strcasecmp() example

```
$var1 = "Hello";
$var2 = "hello";
if (!strcasecmp($var1, $var2)) {
    echo '$var1 is equal to $var2 in a case-insensitive string comparison';
}
```

See also ereg(), strcmp(), substr(), stristr(), strncasecmp(), and strstr().

strchr (PHP 3, PHP 4 >= 4.0.0)

Find the first occurrence of a character

string **strchr** (string haystack, string needle) \linebreak

This function is an alias for strstr(), and is identical in every way.

strcmp (PHP 3, PHP 4 >= 4.0.0)

Binary safe string comparison

int **strcmp** (string str1, string str2) \linebreak

Returns < 0 if `str1` is less than `str2`; > 0 if `str1` is greater than `str2`, and 0 if they are equal.

Note that this comparison is case sensitive.

See also ereg(), strcasecmp(), substr(), stristr(), strncasecmp(), strncmp(), and strstr().

strcoll (PHP 4 >= 4.0.5)

Locale based string comparison

int **strcoll** (string str1, string str2) \linebreak

Returns < 0 if `str1` is less than `str2`; > 0 if `str1` is greater than `str2`, and 0 if they are equal. **strcoll()** uses the current locale for doing the comparisons. If the current locale is C or POSIX, this function is equivalent to strcmp().

Note that this comparison is case sensitive, and unlike strcmp() this function is not binary safe.

See also ereg(), strcmp(), strcasecmp(), substr(), stristr(), strncasecmp(), strncmp(), strstr(), and setlocale().

strcspn (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Find length of initial segment not matching mask

int **strcspn** (string str1, string str2) \linebreak

Returns the length of the initial segment of `str1` which does *not* contain any of the characters in `str2`.

See also strspn().

strip_tags (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Strip HTML and PHP tags from a string

string **strip_tags** (string str [, string allowable_tags]) \linebreak

This function tries to return a string with all HTML and PHP tags stripped from a given `str`. It errors on the side of caution in case of incomplete or bogus tags. It uses the same tag stripping state machine as the fgetss() function.

You can use the optional second parameter to specify tags which should not be stripped.

> **Nota:** `allowable_tags` was added in PHP 3.0.13 and PHP 4.0b3.

Esempio 1. strip_tags() example

```
$string = strip_tags($string, '<a><b><i><u>');
```

Attenzione

This function does not modify any attributes on the tags that you allow using *allowable_tags*, including the `style` and `onmouseover` attributes that a mischievous user may abuse when posting text that will be shown to other users.

stripcslashes (PHP 4 >= 4.0.0)

Un-quote string quoted with addcslashes()

string **stripcslashes** (string str) \linebreak

Returns a string with backslashes stripped off. Recognizes C-like `\n`, `\r` ..., octal and hexadecimal representation.

See also addcslashes().

stripslashes (PHP 3, PHP 4 >= 4.0.0)

Un-quote string quoted with addslashes()

string **stripslashes** (string str) \linebreak

Returns a string with backslashes stripped off. (`\'` becomes `'` and so on.) Double backslashes are made into a single backslash.

See also addslashes().

stristr (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Case-insensitive strstr()

string **stristr** (string haystack, string needle) \linebreak

Returns all of *haystack* from the first occurrence of *needle* to the end. *needle* and *haystack* are examined in a case-insensitive manner.

If *needle* is not found, returns FALSE.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

See also strchr(), strrchr(), substr(), and ereg().

strlen (PHP 3, PHP 4 >= 4.0.0)

Get string length

int **strlen** (string str) \linebreak

Returns the length of *string*.

strnatcasecmp (PHP 4 >= 4.0.0)

Case insensitive string comparisons using a "natural order" algorithm

int **strnatcasecmp** (string str1, string str2) \linebreak

This function implements a comparison algorithm that orders alphanumeric strings in the way a human being would. The behaviour of this function is similar to strnatcmp(), except that the comparison is not case sensitive. For more infomation see: Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) page.

Similar to other string comparison functions, this one returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

See also ereg(), strcasecmp(), substr(), stristr(), strcmp(), strncmp(), strncasecmp(), strnatcmp(), and strstr().

strnatcmp (PHP 4 >= 4.0.0)

String comparisons using a "natural order" algorithm

int **strnatcmp** (string str1, string str2) \linebreak

This function implements a comparison algorithm that orders alphanumeric strings in the way a human being would, this is described as a "natural ordering". An example of the difference between this algorithm and the regular computer string sorting algorithms (used in strcmp()) can be seen below:

```
$arr1 = $arr2 = array("img12.png","img10.png","img2.png","img1.png");
echo "Standard string comparison\n";
usort($arr1,"strcmp");
print_r($arr1);
echo "\nNatural order string comparison\n";
```

```
usort($arr2,"strnatcmp");
print_r($arr2);
```

The code above will generate the following output:

```
Standard string comparison
Array
(
    [0] => img1.png
    [1] => img10.png
    [2] => img12.png
    [3] => img2.png
)

Natural order string comparison
Array
(
    [0] => img1.png
    [1] => img2.png
    [2] => img10.png
    [3] => img12.png
)
```

For more information see: Martin Pool's Natural Order String Comparison (http://naturalordersort.org/) page.

Similar to other string comparison functions, this one returns < 0 if `str1` is less than `str2`; > 0 if `str1` is greater than `str2`, and 0 if they are equal.

Note that this comparison is case sensitive.

See also ereg(), strcasecmp(), substr(), stristr(), strcmp(), strncmp(), strncasecmp(), strnatcasecmp(), strstr(), natsort() and natcasesort().

strncasecmp (PHP 4 >= 4.0.2)

Binary safe case-insensitive string comparison of the first n characters

int **strncasecmp** (string str1, string str2, int len) \linebreak

This function is similar to strcasecmp(), with the difference that you can specify the (upper limit of the) number of characters (`len`) from each string to be used in the comparison. If any of the strings is shorter than `len`, then the length of that string will be used for the comparison.

Returns < 0 if `str1` is less than `str2`; > 0 if `str1` is greater than `str2`, and 0 if they are equal.

See also ereg(), strcasecmp(), strcmp(), substr(), stristr(), and strstr().

strncmp (PHP 4 >= 4.0.0)

Binary safe string comparison of the first n characters

int **strncmp** (string str1, string str2, int len) \linebreak

This function is similar to strcmp(), with the difference that you can specify the (upper limit of the) number of characters (*len*) from each string to be used in the comparison. If any of the strings is shorter than *len*, then the length of that string will be used for the comparison.

Returns < 0 if *str1* is less than *str2*; > 0 if *str1* is greater than *str2*, and 0 if they are equal.

Note that this comparison is case sensitive.

See also ereg(), strncasecmp(), strcasecmp(), substr(), stristr(), strcmp(), and strstr().

strpos (PHP 3, PHP 4 >= 4.0.0)

Find position of first occurrence of a string

int **strpos** (string haystack, string needle [, int offset]) \linebreak

Returns the numeric position of the first occurrence of *needle* in the *haystack* string. Unlike the strrpos(), this function can take a full string as the *needle* parameter and the entire string will be used.

If *needle* is not found, returns FALSE.

> **Nota:** It is easy to mistake the return values for "character found at position 0" and "character not found". Here's how to detect the difference:

```
// in PHP 4.0b3 and newer:
$pos = strpos($mystring, "b");
if ($pos === false) { // note: three equal signs
    // not found...
}

// in versions older than 4.0b3:
$pos = strpos($mystring, "b");
if (!is_integer($pos)) {
    // not found...
}
```

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

The optional *offset* parameter allows you to specify which character in *haystack* to start searching. The position returned is still relative to the the beginning of *haystack*.

See also strrpos(), strrchr(), substr(), stristr(), and strstr().

strrchr (PHP 3, PHP 4 >= 4.0.0)

Find the last occurrence of a character in a string

string **strrchr** (string haystack, string needle) \linebreak

This function returns the portion of *haystack* which starts at the last occurrence of *needle* and goes until the end of *haystack*.

Returns FALSE if *needle* is not found.

If *needle* contains more than one character, the first is used.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

Esempio 1. strrchr() example

```
// get last directory in $PATH
$dir = substr(strrchr($PATH, ":"), 1);

// get everything after last newline
$text = "Line 1\nLine 2\nLine 3";
$last = substr(strrchr($text, 10), 1 );
```

See also strchr(), substr(), stristr(), and strstr().

strrev (PHP 3, PHP 4 >= 4.0.0)

Reverse a string

string **strrev** (string string) \linebreak

Returns *string*, reversed.

Esempio 1. Reversing a string with strrev()

```
<php
echo strrev("Hello world!"); // outputs "!dlrow olleH"
?>
```

strrpos (PHP 3, PHP 4 >= 4.0.0)

Find position of last occurrence of a char in a string

int **strrpos** (string haystack, char needle) \linebreak

Returns the numeric position of the last occurrence of `needle` in the `haystack` string. Note that the needle in this case can only be a single character. If a string is passed as the needle, then only the first character of that string will be used.

If `needle` is not found, returns FALSE.

> **Nota:** It is easy to mistake the return values for "character found at position 0" and "character not found". Here's how to detect the difference:

```
// in PHP 4.0b3 and newer:
$pos = strrpos($mystring, "b");
if ($pos === false) { // note: three equal signs
    // not found...
}

// in versions older than 4.0b3:
$pos = strrpos($mystring, "b");
if (is_string($pos) && !$pos) {
    // not found...
}
```

If `needle` is not a string, it is converted to an integer and applied as the ordinal value of a character.

See also strpos(), strrchr(), substr(), stristr(), and strstr().

strspn (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Find length of initial segment matching mask

int **strspn** (string str1, string str2) \linebreak

Returns the length of the initial segment of `str1` which consists entirely of characters in `str2`.

The line of code:

```
$var = strspn("42 is the answer, what is the question ...", "1234567890");
```

will assign 2 to `$var`, because the string "42" will be the longest segment containing characters from "1234567890".

See also strcspn().

strstr (PHP 3, PHP 4 >= 4.0.0)

Find first occurrence of a string

string **strstr** (string haystack, string needle) \linebreak

Returns part of *haystack* string from the first occurrence of *needle* to the end of *haystack*.

If *needle* is not found, returns FALSE.

If *needle* is not a string, it is converted to an integer and applied as the ordinal value of a character.

> **Nota:** This function is case-sensitive. For case-insensitive searches, use stristr().

Esempio 1. strstr() example

```
$email = 'user@example.com';
$domain = strstr($email, '@');
print $domain; // prints @example.com
```

See also ereg(), preg_match(), strchr(), stristr(), strpos(), strrchr(), and substr().

strtok (PHP 3, PHP 4 >= 4.0.0)

Tokenize string

string **strtok** (string arg1, string arg2) \linebreak

strtok() splits a string (*arg1*) into smaller strings (tokens), with each token being delimited by any character from *arg2*. That is, if you have a string like "This is an example string" you could tokenize this string into its individual words by using the space character as the token.

Esempio 1. strtok() example

```
$string = "This is\tan example\nstring";
/* Use tab and newline as tokenizing characters as well  */
$tok = strtok($string," \n\t");
while ($tok) {
    echo "Word=$tok<br>";
    $tok = strtok(" \n\t");
}
```

Note that only the first call to strtok uses the string argument. Every subsequent call to strtok only needs the token to use, as it keeps track of where it is in the current string. To start over, or to tokenize a new string you simply call strtok with the string argument again to initialize it. Note that you may put multiple tokens in the token parameter. The string will be tokenized when any one of the characters in the argument are found.

The behavior when an empty part was found changed with PHP 4.1.0. The old behavior returned an empty string, while the new, correct, behavior simply skips the part of the string:

Esempio 2. Old strtok() behavior

```
    $first_token  = strtok('/something', '/');
    $second_token = strtok('/');
    var_dump ($first_token, $second_token);

/* Output:
    string(0) ""
    string(9) "something"
*/
```

Esempio 3. New strtok() behavior

```
    $first_token  = strtok('/something', '/');
    $second_token = strtok('/');
    var_dump ($first_token, $second_token);

/* Output:
    string(9) "something"
    bool(false)
*/
```

Also be careful that your tokens may be equal to "0". This evaluates to `FALSE` in conditional expressions.

See also split() and explode().

strtolower (PHP 3, PHP 4 >= 4.0.0)

Make a string lowercase

string **strtolower** (string str) \linebreak

Returns *string* with all alphabetic characters converted to lowercase.

Note that 'alphabetic' is determined by the current locale. This means that in i.e. the default "C" locale, characters such as umlaut-A (Ä) will not be converted.

Esempio 1. strtolower() example

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtolower($str);
print $str; # Prints mary had a little lamb and she loved it so
```

See also strtoupper(), ucfirst(), and ucwords().

strtoupper (PHP 3, PHP 4 >= 4.0.0)

Make a string uppercase

string **strtoupper** (string string) \linebreak

Returns *string* with all alphabetic characters converted to uppercase.

Note that 'alphabetic' is determined by the current locale. For instance, in the default "C" locale characters such as umlaut-a (ä) will not be converted.

Esempio 1. strtoupper() example

```
$str = "Mary Had A Little Lamb and She LOVED It So";
$str = strtoupper($str);
print $str; # Prints MARY HAD A LITTLE LAMB AND SHE LOVED IT SO
```

See also strtolower(), ucfirst(), and ucwords().

strtr (PHP 3, PHP 4 >= 4.0.0)

Translate certain characters

string **strtr** (string str, string from, string to) \linebreak string **strtr** (string str, array replace_pairs) \linebreak

This function returns a copy of *str*, translating all occurrences of each character in *from* to the corresponding character in *to* and returning the result.

If *from* and *to* are different lengths, the extra characters in the longer of the two are ignored.

Esempio 1. strtr() example

```
$addr = strtr($addr, "äåö", "aao");
```

strtr() can be called with only two arguments. If called with two arguments it behaves in a new way: *from* then has to be an array that contains string -> string pairs that will be replaced in the source string. **strtr()** will always look for the longest possible match first and will *NOT* try to replace stuff that it has already worked on.

Examples:

```
$trans = array("hello" => "hi", "hi" => "hello");
echo strtr("hi all, I said hello", $trans) . "\n";
```

This will show: "hello all, I said hi",

> **Nota:** This optional *to* and *from* parameters were added in PHP 4.0.0

See also ereg_replace().

substr (PHP 3, PHP 4 >= 4.0.0)

Return part of a string

string **substr** (string string, int start [, int length]) \linebreak

Substr returns the portion of *string* specified by the *start* and *length* parameters.

If *start* is positive, the returned string will start at the *start*'th position in *string*, counting from zero. For instance, in the string '`abcdef`', the character at position `0` is '`a`', the character at position `2` is '`c`', and so forth.

Esempio 1. Basic substr() usage

```
$rest = substr("abcdef", 1);    // returns "bcdef"
$rest = substr("abcdef", 1, 3); // returns "bcd"
$rest = substr("abcdef", 0, 4); // returns "abcd"
$rest = substr("abcdef", 0, 8); // returns "abcdef"
```

If *start* is negative, the returned string will start at the *start*'th character from the end of *string*.

Esempio 2. Using a negative *start*

```
$rest = substr("abcdef", -1);    // returns "f"
$rest = substr("abcdef", -2);    // returns "ef"
$rest = substr("abcdef", -3, 1); // returns "d"
```

If *length* is given and is positive, the string returned will contain at most *length* characters beginning from *start* (depending on the length of *string*. If *string* is less than *start* characters long, FALSE will be returned.

If *length* is given and is negative, then that many characters will be omitted from the end of *string* (after the start position has been calculated when a *start* is negative). If *start* denotes a position beyond this truncation, an empty string will be returned.

Esempio 3. Using a negative *length*

```
$rest = substr("abcdef", 0, -1);  // returns "abcde"
$rest = substr("abcdef", 2, -1);  // returns "cde"
$rest = substr("abcdef", 4, -4);  // returns ""
$rest = substr("abcdef", -3, -1); // returns "de"
```

See also strrchr() and ereg().

substr_count (PHP 4 >= 4.0.0)

Count the number of substring occurrences

int **substr_count** (string haystack, string needle) \linebreak

substr_count() returns the number of times the *needle* substring occurs in the *haystack* string.

Esempio 1. substr_count() example

```
print substr_count("This is a test", "is"); // prints out 2
```

substr_replace (PHP 4 >= 4.0.0)

Replace text within a portion of a string

string **substr_replace** (string string, string replacement, int start [, int length]) \linebreak

substr_replace() replaces a copy of *string* delimited by the *start* and (optionally) *length* parameters with the string given in *replacement*. The result is returned.

If *start* is positive, the replacing will begin at the *start*'th offset into *string*.

If *start* is negative, the replacing will begin at the *start*'th character from the end of *string*.

If *length* is given and is positive, it represents the length of the portion of *string* which is to be replaced. If it is negative, it represents the number of characters from the end of *string* at which to stop replacing. If it is not given, then it will default to strlen(*string*); i.e. end the replacing at the end of *string*.

Esempio 1. substr_replace() example

```
<?php
$var = 'ABCDEFGH:/MNRPQR/';
echo "Original: $var<hr>\n";

/* These two examples replace all of $var with 'bob'. */
echo substr_replace($var, 'bob', 0) . "<br>\n";
echo substr_replace($var, 'bob', 0, strlen($var)) . "<br>\n";

/* Insert 'bob' right at the beginning of $var. */
echo substr_replace($var, 'bob', 0, 0) . "<br>\n";

/* These next two replace 'MNRPQR' in $var with 'bob'. */
echo substr_replace($var, 'bob', 10, -1) . "<br>\n";
echo substr_replace($var, 'bob', -7, -1) . "<br>\n";

/* Delete 'MNRPQR' from $var. */
echo substr_replace($var, ", 10, -1) . "<br>\n";
?>
```

See also str_replace() and substr().

trim (PHP 3, PHP 4 >= 4.0.0)

Strip whitespace from the beginning and end of a string

string **trim** (string str [, string charlist]) \linebreak

> **Nota:** The optional *charlist* parameter was added in PHP 4.1.0

This function returns a string with whitespace stripped from the beginning and end of *str*. Without the second parameter, **trim()** will strip these characters:

- " " (ASCII 32 (0x20)), an ordinary space.

- "\t" (ASCII `9` (`0x09`)), a tab.
- "\n" (ASCII `10` (`0x0A`)), a new line (line feed).
- "\r" (ASCII `13` (`0x0D`)), a carriage return.
- "\0" (ASCII `0` (`0x00`)), the `NUL`-byte.
- "\x0B" (ASCII `11` (`0x0B`)), a vertical tab.

You can also specify the characters you want to strip, by means of the *`charlist`* parameter. Simply list all characters that you want to be stripped. With `..` you can specify a range of characters.

Esempio 1. Usage example of trim()

```
<?php

$text = "\t\tThese are a few words :) ...  ";
$trimmed = trim($text);
// $trimmed = "These are a few words :) ..."
$trimmed = trim($text," \t.");
// $trimmed = "These are a few words :)"
$clean = trim($binary,"\0x00..\0x1F");
// trim the ASCII control characters at the beginning and end of $binary
// (from 0 to 31 inclusive)

?>
```

See also ltrim() and rtrim().

ucfirst (PHP 3, PHP 4 >= 4.0.0)

Make a string’s first character uppercase

string **ucfirst** (string str) \linebreak

Returns a string with the first character of *`str`* capitalized, if that character is alphabetic.

Note that ’alphabetic’ is determined by the current locale. For instance, in the default "C" locale characters such as umlaut-a (ä) will not be converted.

Esempio 1. ucfirst() example

```
$foo = ’hello world!’;
$foo = ucfirst($foo);             // Hello world!

$bar = ’HELLO WORLD!’;
$bar = ucfirst($bar);             // HELLO WORLD!
$bar = ucfirst(strtolower($bar)); // Hello world!
```

See also strtolower(), strtoupper(), and ucwords().

ucwords (PHP 3>= 3.0.3, PHP 4 >= 4.0.0)

Uppercase the first character of each word in a string

string **ucwords** (string str) \linebreak

Returns a string with the first character of each word in *str* capitalized, if that character is alphabetic.

Esempio 1. ucwords() example

```
$foo = 'hello world!';
$foo = ucwords($foo);             // Hello World!

$bar = 'HELLO WORLD!';
$bar = ucwords($bar);             // HELLO WORLD!
$bar = ucwords(strtolower($bar)); // Hello World!
```

Nota: The definition of a word is any string of characters that is immediately after a whitespace (These are: space, form-feed, newline, carriage return, horizontal tab, and vertical tab).

See also strtoupper(), strtolower() and ucfirst().

vprintf (PHP 4 >= 4.1.0)

Output a formatted string

void **vprintf** (string format, array args) \linebreak

Display array values as a formatted string according to *format* (which is described in the documentation for sprintf()).

Operates as printf() but accepts an array of arguments, rather than a variable number of arguments.

See also: printf(), sprintf(), vsprintf()

vsprintf (PHP 4 >= 4.1.0)

Return a formatted string

string **vsprintf** (string format, array args) \linebreak

Return array values as a formatted string according to *format* (which is described in the documentation for sprintf()).

Operates as sprintf() but accepts an array of arguments, rather than a variable number of arguments.

See also: sprintf(), **vsprintf()**, vprintf()

wordwrap (PHP 4 >= 4.0.2)

Wraps a string to a given number of characters using a string break character.

string **wordwrap** (string str [, int width [, string break [, int cut]]]) \linebreak

Returns a string with *str* wrapped at the column number specified by the (optional) *width* parameter. The line is broken using the (optional) *break* parameter.

wordwrap() will automatically wrap at column 75 and break using '\n' (newline) if *width* or *break* are not given.

If the *cut* is set to 1, the string is always wrapped at the specified width. So if you have a word that is larger than the given width, it is broken apart. (See second example).

> **Nota:** The optional *cut* parameter was added in PHP 4.0.3

Esempio 1. wordwrap() example

```
$text = "The quick brown fox jumped over the lazy dog.";
$newtext = wordwrap( $text, 20 );

echo "$newtext\n";
```

This example would display:

```
The quick brown fox
jumped over the lazy dog.
```

Esempio 2. wordwrap() example

```
$text = "A very long woooooooooooord.";
$newtext = wordwrap( $text, 8, "\n", 1);

echo "$newtext\n";
```

This example would display:

```
A very
long
wooooooo
ooooord.
```

See also nl2br().

XCVII. Sybase functions

sybase_affected_rows (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

get number of affected rows in last query

int **sybase_affected_rows** ([int link_identifier]) \linebreak

Returns: The number of affected rows by the last query.

sybase_affected_rows() returns the number of rows affected by the last INSERT, UPDATE or DELETE query on the server associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

This command is not effective for SELECT statements, only on statements which modify records. To retrieve the number of rows returned from a SELECT, use sybase_num_rows().

> **Nota:** This function is only available using the CT library interface to Sybase, and not the DB library.

sybase_close (PHP 3, PHP 4 >= 4.0.0)

close Sybase connection

bool **sybase_close** (int link_identifier) \linebreak

Returns: TRUE on success, FALSE on error

sybase_close() closes the link to a Sybase database that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed.

Note that this isn't usually necessary, as non-persistent open links are automatically closed at the end of the script's execution.

sybase_close() will not close persistent links generated by sybase_pconnect().

See also: sybase_connect(), sybase_pconnect().

sybase_connect (PHP 3, PHP 4 >= 4.0.0)

open Sybase server connection

int **sybase_connect** (string servername, string username, string password [, string charset]) \linebreak

Returns: A positive Sybase link identifier on success, or FALSE on error.

sybase_connect() establishes a connection to a Sybase server. The servername argument has to be a valid servername that is defined in the 'interfaces' file.

In case a second call is made to **sybase_connect()** with the same arguments, no new link will be established, but instead, the link identifier of the already opened link will be returned.

The link to the server will be closed as soon as the execution of the script ends, unless it's closed earlier by explicitly calling sybase_close().

See also sybase_pconnect(), sybase_close().

sybase_data_seek (PHP 3, PHP 4 >= 4.0.0)

move internal row pointer

bool **sybase_data_seek** (int result_identifier, int row_number) \linebreak

Returns: TRUE on success, FALSE on failure

sybase_data_seek() moves the internal row pointer of the Sybase result associated with the specified result identifier to pointer to the specifyed row number. The next call to sybase_fetch_row() would return that row.

See also: **sybase_data_seek()**.

sybase_fetch_array (PHP 3, PHP 4 >= 4.0.0)

fetch row as array

array **sybase_fetch_array** (int result) \linebreak

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

sybase_fetch_array() is an extended version of sybase_fetch_row(). In addition to storing the data in the numeric indices of the result array, it also stores the data in associative indices, using the field names as keys.

An important thing to note is that using **sybase_fetch_array()** is NOT significantly slower than using sybase_fetch_row(), while it provides a significant added value.

For further details, also see sybase_fetch_row().

sybase_fetch_field (PHP 3, PHP 4 >= 4.0.0)

get field information

object **sybase_fetch_field** (int result [, int field_offset]) \linebreak

Returns an object containing field information.

sybase_fetch_field() can be used in order to obtain information about fields in a certain query result. If the field offset isn't specified, the next field that wasn't yet retreived by **sybase_fetch_field()** is retreived.

The properties of the object are:

- name - column name. if the column is a result of a function, this property is set to computed#N, where #N is a serial number.

- column_source - the table from which the column was taken
- max_length - maximum length of the column
- numeric - 1 if the column is numeric
- type - datatype of the column

See also sybase_field_seek()

sybase_fetch_object (PHP 3, PHP 4 >= 4.0.0)

fetch row as object

int **sybase_fetch_object** (int result) \linebreak

Returns: An object with properties that correspond to the fetched row, or FALSE if there are no more rows.

sybase_fetch_object() is similar to sybase_fetch_array(), with one difference - an object is returned, instead of an array. Indirectly, that means that you can only access the data by the field names, and not by their offsets (numbers are illegal property names).

Speed-wise, the function is identical to sybase_fetch_array(), and almost as quick as sybase_fetch_row() (the difference is insignificant).

See also: sybase_fetch_array() and sybase_fetch_row().

sybase_fetch_row (PHP 3, PHP 4 >= 4.0.0)

get row as enumerated array

array **sybase_fetch_row** (int result) \linebreak

Returns: An array that corresponds to the fetched row, or FALSE if there are no more rows.

sybase_fetch_row() fetches one row of data from the result associated with the specified result identifier. The row is returned as an array. Each result column is stored in an array offset, starting at offset 0.

Subsequent call to **sybase_fetch_row()** would return the next row in the result set, or FALSE if there are no more rows.

See also: sybase_fetch_array(), sybase_fetch_object(), sybase_data_seek(), **sybase_fetch_lengths()**, and sybase_result().

sybase_field_seek (PHP 3, PHP 4 >= 4.0.0)

set field offset

int **sybase_field_seek** (int result, int field_offset) \linebreak

Seeks to the specified field offset. If the next call to sybase_fetch_field() won't include a field offset, this field would be returned.

See also: sybase_fetch_field().

sybase_free_result (PHP 3, PHP 4 >= 4.0.0)

free result memory

bool **sybase_free_result** (int result) \linebreak

sybase_free_result() only needs to be called if you are worried about using too much memory while your script is running. All result memory will automatically be freed when the script ends. You may call **sybase_free_result()** with the result identifier as an argument and the associated result memory will be freed.

sybase_get_last_message (PHP 3, PHP 4 >= 4.0.0)

Returns the last message from the server

string **sybase_get_last_message** (void) \linebreak

sybase_get_last_message() returns the last message reported by the server.

sybase_min_client_severity (PHP 3, PHP 4 >= 4.0.0)

Sets minimum client severity

void **sybase_min_client_severity** (int severity) \linebreak

sybase_min_client_severity() sets the minimum client severity level.

> **Nota:** This function is only available using the CT library interface to Sybase, and not the DB library.

See also: sybase_min_server_severity().

sybase_min_error_severity (PHP 3, PHP 4 >= 4.0.0)

Sets minimum error severity

void **sybase_min_error_severity** (int severity) \linebreak

sybase_min_error_severity() sets the minimum error severity level.

See also: sybase_min_message_severity().

sybase_min_message_severity (PHP 3, PHP 4 >= 4.0.0)

Sets minimum message severity

void **sybase_min_message_severity** (int severity) \linebreak

sybase_min_message_severity() sets the minimum message severity level.

See also: sybase_min_error_severity().

sybase_min_server_severity (PHP 3, PHP 4 >= 4.0.0)

Sets minimum server severity

void **sybase_min_server_severity** (int severity) \linebreak

sybase_min_server_severity() sets the minimum server severity level.

> **Nota:** This function is only available using the CT library interface to Sybase, and not the DB library.

See also: sybase_min_client_severity().

sybase_num_fields (PHP 3, PHP 4 >= 4.0.0)

get number of fields in result

int **sybase_num_fields** (int result) \linebreak

sybase_num_fields() returns the number of fields in a result set.

See also: **sybase_db_query()**, sybase_query(), sybase_fetch_field(), sybase_num_rows().

sybase_num_rows (PHP 3, PHP 4 >= 4.0.0)

get number of rows in result

int **sybase_num_rows** (int result) \linebreak

sybase_num_rows() returns the number of rows in a result set.

See also: **sybase_db_query()**, sybase_query() and, sybase_fetch_row().

sybase_pconnect (PHP 3, PHP 4 >= 4.0.0)

open persistent Sybase connection

int **sybase_pconnect** (string servername, string username, string password [, string charset]) \linebreak

Returns: A positive Sybase persistent link identifier on success, or FALSE on error

sybase_pconnect() acts very much like sybase_connect() with two major differences.

First, when connecting, the function would first try to find a (persistent) link that's already open with the same host, username and password. If one is found, an identifier for it will be returned instead of opening a new connection.

Second, the connection to the SQL server will not be closed when the execution of the script ends. Instead, the link will remain open for future use (sybase_close() will not close links established by **sybase_pconnect()()**).

This type of links is therefore called 'persistent'.

sybase_query (PHP 3, PHP 4 >= 4.0.0)

send Sybase query

int **sybase_query** (string query, int link_identifier) \linebreak

Returns: A positive Sybase result identifier on success, or FALSE on error.

sybase_query() sends a query to the currently active database on the server that's associated with the specified link identifier. If the link identifier isn't specified, the last opened link is assumed. If no link is open, the function tries to establish a link as if sybase_connect() was called, and use it.

See also: **sybase_db_query()**, sybase_select_db(), and sybase_connect().

sybase_result (PHP 3, PHP 4 >= 4.0.0)

get result data

string **sybase_result** (int result, int row, mixed field) \linebreak

Returns: The contents of the cell at the row and offset in the specified Sybase result set.

sybase_result() returns the contents of one cell from a Sybase result set. The field argument can be the field's offset, or the field's name, or the field's table dot field's name (tablename.fieldname). If the column name has been aliased ('select foo as bar from...'), use the alias instead of the column name.

When working on large result sets, you should consider using one of the functions that fetch an entire row (specified below). As these functions return the contents of multiple cells in one function call, they're MUCH quicker than sybase_result(). Also, note that specifying a numeric offset for the field argument is much quicker than specifying a fieldname or tablename.fieldname argument.

Recommended high-performance alternatives: sybase_fetch_row(), sybase_fetch_array(), and sybase_fetch_object().

sybase_select_db (PHP 3, PHP 4 >= 4.0.0)

select Sybase database

bool **sybase_select_db** (string database_name, int link_identifier) \linebreak

Returns: TRUE on success, FALSE on error

sybase_select_db() sets the current active database on the server that's associated with the specified link identifier. If no link identifier is specified, the last opened link is assumed. If no link is open, the function will try to establish a link as if sybase_connect() was called, and use it.

Every subsequent call to sybase_query() will be made on the active database.

See also: sybase_connect(), sybase_pconnect(), and sybase_query()

XCVIII. URL Functions

base64_decode (PHP 3, PHP 4 >= 4.0.0)

Decodes data encoded with MIME base64

string **base64_decode** (string encoded_data) \linebreak

base64_decode() decodes *encoded_data* and returns the original data. The returned data may be binary.

See also: base64_encode(), RFC2045 section 6.8.

base64_encode (PHP 3, PHP 4 >= 4.0.0)

Encodes data with MIME base64

string **base64_encode** (string data) \linebreak

base64_encode() returns *data* encoded with base64. This encoding is designed to make binary data survive transport through transport layers that are not 8-bit clean, such as mail bodies.

Base64-encoded data takes about 33% more space than the original data.

See also: base64_decode(), chunk_split(), RFC2045 section 6.8.

parse_url (PHP 3, PHP 4 >= 4.0.0)

Parse a URL and return its components

array **parse_url** (string url) \linebreak

This function returns an associative array returning any of the various components of the URL that are present. This includes the

- *scheme* - e.g. http
- *host*
- *port*
- *user*
- *pass*
- *path*
- *query* - after the question mark ?
- *fragment* - after the hashmark #

See also pathinfo().

rawurldecode (PHP 3, PHP 4 >= 4.0.0)

Decode URL-encoded strings

string **rawurldecode** (string str) \linebreak

Returns a string in which the sequences with percent (%) signs followed by two hex digits have been replaced with literal characters. For example, the string

```
foo%20bar%40baz
```

decodes into

```
foo bar@baz
```

.

> **Nota: rawurldecode()** does not decode plus symbols ('+') into spaces. urldecode() does.

See also rawurlencode(), urldecode(), urlencode().

rawurlencode (PHP 3, PHP 4 >= 4.0.0)

URL-encode according to RFC1738

string **rawurlencode** (string str) \linebreak

Returns a string in which all non-alphanumeric characters except

```
-_.
```

have been replaced with a percent (%) sign followed by two hex digits. This is the encoding described in RFC1738 for protecting literal characters from being interpreted as special URL delimiters, and for protecting URL's from being mangled by transmission media with character conversions (like some email systems). For example, if you want to include a password in an FTP URL:

Esempio 1. rawurlencode() example 1

```
echo '<a href="ftp://user:', rawurlencode('foo @+%/'),
     '@ftp.my.com/x.txt">';
```

Or, if you pass information in a PATH_INFO component of the URL:

Esempio 2. rawurlencode() example 2

```
echo '<a href="http://x.com/department_list_script/',
    rawurlencode('sales and marketing/Miami'), '">';
```

See also rawurldecode(), urldecode(), urlencode().

urldecode (PHP 3, PHP 4 >= 4.0.0)

Decodes URL-encoded string

string **urldecode** (string str) \linebreak

Decodes any `%##` encoding in the given string. The decoded string is returned.

Esempio 1. urldecode() example

```
$a = explode('&', $QUERY_STRING);
$i = 0;
while ($i < count($a)) {
    $b = split('=', $a[$i]);
    echo 'Value for parameter ', htmlspecialchars(urldecode($b[0])),
         ' is ', htmlspecialchars(urldecode($b[1])), "<br />\n";
    $i++;
}
```

See also urlencode(), rawurlencode(), rawurldecode().

urlencode (PHP 3, PHP 4 >= 4.0.0)

URL-encodes string

string **urlencode** (string str) \linebreak

Returns a string in which all non-alphanumeric characters except `-_.` have been replaced with a percent (`%`) sign followed by two hex digits and spaces encoded as plus (`+`) signs. It is encoded the same way that the posted data from a WWW form is encoded, that is the same way as in `application/x-www-form-urlencoded` media type. This differs from the RFC1738 encoding (see rawurlencode()) in that for historical reasons, spaces are encoded as plus (+) signs. This function is convenient when encoding a string to be used in a query part of an URL, as a convenient way to pass variables to the next page:

Esempio 1. urlencode() example

```
echo '<a href="mycgi?foo=', urlencode($userinput), '">';
```

Note: Be careful about variables that may match HTML entities. Things like &, © and £ are parsed by the browser and the actual entity is used instead of the desired variable name. This is an obvious hassle that the W3C has been telling people about for years. The reference is here: http://www.w3.org/TR/html4/appendix/notes.html#h-B.2.2 PHP supports changing the argument separator to the W3C-suggested semi-colon through the arg_separator .ini directive. Unfortunately most user agents do not send form data in this semi-colon separated format. A more portable way around this is to use & instead of & as the separator. You don't need to change PHP's arg_separator for this. Leave it as &, but simply encode your URLs using htmlentities()(urlencode($data)).

Esempio 2. urlencode/htmlentities() example

```
echo '<a href="mycgi?foo=', htmlentities(urlencode($userinput)), '">';
```

See also urldecode(), htmlentities(), rawurldecode(), rawurlencode().

XCIX. Funzioni di Variabili

Per maggiori informazioni sul comportamento delle variabili vedere la sezione Variabili nel capitolo Riferimento al Linguaggio del manuale.

doubleval (PHP 3, PHP 4 >= 4.0.0)

Alias di floatval()

Questa funzione è un alias di floatval().

> **Nota:** Questo alias è una rimanenza dalla ridenominazione di alcune funzioni. Nelle vecchie versioni di PHP, occorre utilizzare questo alias al posto della funzione floatval(), questo perchè la suddetta funzione non era disponibile.

empty (unknown)

Determina se una variabile è valorizzata

boolean **empty** (mixed var) \linebreak

> **Nota: empty()** è un costrutto del linguaggio.

Questo è l'opposto di `(boolean)` *`var`*, tranne che non viene dato alcun warning quando la variabile non è valorizzata. Vedere il capitolo Conversione a booleano per maggiori informazioni.

```
$var = 0;
if (empty($var)) { // restituisce true
    print '$var è uguale a 0 oppure non è definita';
}
if (!isset($var)) { // restituisce false
    print '$var non è definita';
}
```

Si noti che la funzione perde di significato se applicata a qualcosa che non sia una variabile; ad esempio **empty (addslashes ($name))** non ha significato perchè tenta di verificare se un qualcosa che non è una variabile è una variabile con un valore `FALSE`.

Vedere anche isset() e unset().

floatval (PHP 4 >= 4.2.0)

Restituisce il valore di una variabile di tipo float

float **floatval** (mixed var) \linebreak

La funzione restituisce il valore di tipo float di *`var`*.

La variabile *`Var`* può anche essere di tipo scalare. La funzione **floatval()** non può essere usata con variabili di tipo array oppure object.

```
$var = '122.34343The';
$valore_float_di_var = floatval ($var);
print $valore_float_di_var; // visualizza 122.34343
```

Vedere anche intval(), strval(), settype() e Manipolazione dei Tipi .

get_defined_vars (PHP 4 >= 4.0.4)

Restituisce un'array contenente tutte le variabili definite

array **get_defined_vars** (void) \linebreak

Questa funzione restituisce un'array multidimensionale contenente la lista di tutte le variabili definite, siano esse d'ambiente, variabili del server o definite dall'utente.

```
$b = array(1,1,2,3,5,8);

$arr = get_defined_vars();

// visualizza $b
print_r($arr["b"]);

// visualizza il percorso dell'interprete PHP (se usato come CGI)
// ad esempio /usr/local/bin/php
echo $arr["_"];

// visualizza i parametri della linea di comando (se presenti)
print_r($arr["argv"]);

// visualizza tutte le variabili del server
print_r($arr["HTTP_SERVER_VARS"]);

// visualizza tutte le chiavi disponibili nell'array delle variabili
print_r(array_keys(get_defined_vars()));
```

Vedere anche get_defined_functions() e get_defined_constants().

get_resource_type (PHP 4 >= 4.0.2)

Restituisce il tipo di risorsa

string **get_resource_type** (resource handle) \linebreak

Questa funzione restituisce una stringa rappresentante il tipo di risorsa passato. Se il parametro passato non è una risorsa valida la funzione genera un errore.

```
$c = mysql_connect();
echo get_resource_type($c)."\n";
// visualizza: mysql link

$fp = fopen("foo","w");
echo get_resource_type($fp)."\n";
// visualizza: file

$doc = new_xmldoc("1.0");
echo get_resource_type($doc->doc)."\n";
// visualizza: domxml document
```

gettype (PHP 3, PHP 4 >= 4.0.0)

Resituisce il tipo di una variabile

string **gettype** (mixed var) \linebreak

Restituisce il tipo della variabile PHP *var*.

> **Attenzione**
>
> Non utilizzare **gettype()** per testare certi tipi di variabili poichè la stringa restistuita potrebbe variare nelle versioni future. Inoltre questa funzione è lenta dato che richiede confronti tra stringhe .
>
> Piuttosto utilizzare le funzioni `is_*`.

La stringa restituita può assumere i seguenti valori:

- "boolean" (dalla versione 4 di PHP)
- "integer"
- "double" (per ragioni storiche viene restituito "double" in caso di variabile di tipo float, e non semplicemente "float")
- "string"
- "array"
- "object"
- "resource" (dalla versione 4 di PHP)
- "NULL" (dalla versione 4 di PHP)
- "user function" (soltanto in PHP 3, deprecated)

- "unknown type"

Nella versione 4 di PHP si dovrebbe applicare alle funzioni function_exists() o method_exists() al posto del precedente **gettype()**.

Vedere anche settype(), is_array(), is_bool(), is_float(), is_integer(), is_null(), is_numeric(), is_object(), is_resource(), is_scalar() e is_string().

import_request_variables (PHP 4 >= 4.1.0)

Imposta la visibiltà a globale per le variabili GET/POST/Cookie

bool **import_request_variables** (string tipo [, string prefisso]) \linebreak

Imposta la visibilità delle variabili GET/POST/Cookie a globale. Ciò risulta utile nei casi in cui si è disabilitato register_globals, ma si vuole avere per qualche variabile una visibilità globale.

Tramite il parametro *tipo*, si può specificare quale variabile rendere visibile. I valori ammessi sono i caratteri ’G’, ’P’ e ’C’ rispettivamente per GET, POST e Cookie. Questi caratteri non distinguono tra maiuscole e minuscole pertanto si può usare qualsiasi combinazione di ’g’, ’p’ e ’c’. Occorre prestare attenzione all’ordine delle lettere, ad esempio usando "gp", le variabili POST sovrascrivono le variabili GET con il medesimo nome. Qualsiasi altra lettera al di fuori di GPC sarà scartata.

Nota: Sebbene il parametro *prefisso* sia opzionale, si ottiene un errore di livello "notice" se non si specifica il prefisso, o si indica una stringa vuota. Ciò può comportare dei rischi di sicurezza. Gli errori di livello "notice" non sono visualizzati con la soglia minima di errore standard.

```
// Questo esempio rende visibili le variabili GET e POST
// con il prefisso "rvar_"
import_request_variables("gP", "rvar_");
```

Vedere anche register_globals e track_vars.

intval (PHP 3, PHP 4 >= 4.0.0)

Estrae il valore intero da una variabile

int **intval** (mixed var [, int base]) \linebreak

Estrae il valore intero di *var*, utilizzando la base definita a parametro per la conversione. (La base vale 10 di default).

Var può essere una qualsiasi variabile scalare. Non è possibile utilizzare **intval()** con variabili di tipo array o object.

Nota: Il parametro *base* di **intval()** non ha effetto se *var* non è una stringa.

Vedere anche doubleval(), strval(), settype() e Manipolazione dei Tipi.

is_array (PHP 3, PHP 4 >= 4.0.0)

Verifica se una variabile è un array

bool **is_array** (mixed var) \linebreak

Restituisce TRUE se *var* è un array, FALSE in caso contrario.

Vedere anche is_float(), is_int(), is_integer(), is_string() e is_object().

is_bool (PHP 4 >= 4.0.0)

Verifica se una variabile è di tipo boolean

bool **is_bool** (mixed var) \linebreak

Restituisce TRUE se *var* è di tipo boolean.

Vedere anche is_array(), is_float(), is_int(), is_integer(), is_string(), and is_object().

is_callable (PHP 4 >= 4.0.6)

Verifica se l'argomento è un valido costrutto richiamabile.

bool **is_callable** (mixed var [, bool syntax_only [, string callable_name]]) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

is_double (PHP 3, PHP 4 >= 4.0.0)

Alias di is_float()

Questa funzione è un alias di is_float().

is_float (PHP 3, PHP 4 >= 4.0.0)

Verifica se una variabile è di tipo float (decimale a virgola mobile)

bool **is_float** (mixed var) \linebreak

Questa funzione restituisce TRUE se *var* è di tipo float, FALSE in caso contrario.

> **Nota:** Per verificare se una variabile è un numero oppure una stringa numerica (come le variabili dei form, che sono sempre stringhe) occorre usare la funzione is_numeric().

Vedere anche is_bool(), is_int(), is_integer(), is_numeric(), is_string(), is_array() e is_object(),

is_int (PHP 3, PHP 4 >= 4.0.0)

Verifica se una variabile è di tipo integer

bool **is_int** (mixed var) \linebreak

Restituisce TRUE se *var* è di tipo integer, FALSE in caso contrario.

> **Nota:** Per verificare se una variabile è un numero oppure una stringa numerica (come le variabili dei form, che sono sempre stringhe) occorre usare la funzione is_numeric().

Vedere anche is_bool(), is_float(), is_integer(), is_numeric(), is_string(), is_array() e is_object().

is_integer (PHP 3, PHP 4 >= 4.0.0)

Alias di is_int()

Questa funzione è un alias di is_int().

is_long (PHP 3, PHP 4 >= 4.0.0)

Alias di is_int()

Questa funzione è un alias di is_int().

is_null (PHP 4 >= 4.0.4)

Verifica se la variabile è di tipo `NULL`

bool **is_null** (mixed var) \linebreak

Restituisce `TRUE` se *var* è di tipo null, `FALSE` in caso contrario.

Vedere anche is_bool(), is_numeric(), is_float(), is_int(), is_string(), is_object() e is_array().

is_numeric (PHP 4 >= 4.0.0)

Verifica se una variabile è un numero o una stringa numerica

bool **is_numeric** (mixed var) \linebreak

Restituisce `TRUE` se *var* è un numero o una stringa numerica, `FALSE` in caso contrario.

Vedere anche is_bool(), is_float(), is_int(), is_string(), is_object(), is_array() e is_integer().

is_object (PHP 3, PHP 4 >= 4.0.0)

Verifica se una variabile è di tipo object

bool **is_object** (mixed var) \linebreak

Restituisce `TRUE` se *var* è un di tipo object, `FALSE` in caso contrario.

Vedere anche is_bool(), is_int(), is_integer(), is_float(), is_string(), and is_array().

is_real (PHP 3, PHP 4 >= 4.0.0)

Alias di is_float()

Questa funzione è un alias di is_float().

is_resource (PHP 4 >= 4.0.0)

Verifica se una variabile è una risorsa

bool **is_resource** (mixed var) \linebreak

is_resource() restituisce `TRUE` se la variabile individuata dal parametro *var* è di tipo resource, altrimenti restituisce `FALSE`.

Per maggiori dettagli fare riferimento alla documentazione di resource-type

is_scalar (PHP 4 >= 4.0.5)

Verifica se la variabile è di tipo scalare

bool **is_scalar** (mixed var) \linebreak

La funzione **is_scalar()** restituisce `TRUE` se la variabile indicata dal parametro *var* è di tipo scalare, in caso contrario restituisce `FALSE`.

Le variabili scalari sono quelle contenenti valori di tipo integer, float, string oppure boolean. I tipi array, object e resource non sono scalari.

```
function show_var($var) {
    if (is_scalar($var)) {
        echo $var;
    } else {
        var_dump($var);
    }
}
$pi = 3.1416;
$proteins = array("hemoglobin", "cytochrome c oxidase", "ferredoxin");

show_var($pi);
// visualizza: 3.1416

show_var($proteins)
// visualizza:
// array(3) {
//   [0]=>
//   string(10) "hemoglobin"
//   [1]=>
//   string(20) "cytochrome c oxidase"
//   [2]=>
//   string(10) "ferredoxin"
// }
```

Nota: La funzione **is_scalar()** non considera il tipo resource come valore scalare, dato che il tipo resource è una tipologia di dato astratto che attualmente si basa su interi. Tuttavia non ci si può basare su questo tipo di implementazione, in futuro potrebbe cambiare.

Vedere anche is_bool(), is_numeric(), is_float(), is_int(), is_real(), is_string(), is_object(), is_array() e is_integer().

is_string (PHP 3, PHP 4 >= 4.0.0)

Verifica se una variabile è una stringa

bool **is_string** (mixed var) \linebreak

Restituisce `TRUE` se *var* è di tipo string, `FALSE` in caso contrario.

Vedere anche is_bool(), is_int(), is_integer(), is_float(), is_real(), is_object() e is_array().

isset (unknown)

Verifica se una variabile è definita

boolean **isset** (mixed var [, ...]) \linebreak

> **Nota: isset()** è un costrutto del linguaggio.

Restituisce `TRUE` se *var* esiste; `FALSE` in caso contrario.

Se una variabile è stata cancellata con unset(), non potrà essere **isset()**. La funzione **isset()** restituirà `FALSE` se viene utilizzata per testare una variabile valorizzata a `NULL`. Inoltre occorre notare che il byte `NULL` (`"\0"`) non equivale alla costante PHP `NULL`.

```
$a = "test";
$b = "anothertest";

echo isset ($a); // TRUE
echo isset ($a, $b) //TRUE

unset ($a);
echo isset ($a); // FALSE
echo isset ($a, $b); //FALSE

$foo = NULL;
print isset ($foo); // FALSE
```

Vedere anche empty() e unset().

print_r (PHP 4 >= 4.0.0)

Stampa informazioni relative al contenuto di una variabile in formato leggibile

void **print_r** (mixed expression) \linebreak

Questa funzione stampa delle informazioni sul contenuto di una variabile in un formato facilmente leggibile. Se la variabile contiene una stringa, un intero o un numero decimale, il valore stesso viene visualizzato. Se la variabile contiene un array i valori vengono visualizzati in un formato che evidenzia le chiavi ed i relativi elementi. Una notazione simile viene utilizzata per gli oggetti.

Occorre ricordarsi che **print_r()** posiziona il puntatore dell'array alla fine. Pertanto utilizzare reset() per riportarsi all'inizio.

> **Suggerimento:** Come con qualsiasi cosa che invia il risultato direttamente al browser, è possibile utilizzare la funzione output-control per catturare l'uscita di questa funzione e salvarla - per esempio - in una stringa.

```
<pre>
<?php
    $a = array ('a' => 'apple', 'b' => 'banana', 'c' => array ('x','y','z'));
    print_r ($a);
?>
</pre>
```

Il precedente esempio visualizzerà:

```
<pre>
Array
(
    [a] => apple
    [b] => banana
    [c] => Array
        (
            [0] => x
            [1] => y
            [2] => z
        )
)
</pre>
```

> **Nota:** Questa funzione continua all'infinito se riceve come parametro un vettore o un oggetto contenente un riferimento diretto od indiretto a se stesso oppure contenente ulteriori vettori o oggetti che a loro volta referenziano il padre o se stessi. Un caso evidente è `print_r($GLOBALS)`, in quanto `$GLOBALS` è a sua volta una variabile globale e in quanto tale contiene una referenza a se stessa.

Vedere anche ob_start(), var_dump() e var_export().

serialize (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Genera una versione archiviabile del valore

string **serialize** (mixed value) \linebreak

La funzione **serialize()** restituisce una stringa contenente un flusso di bytes rappresentante `value` che possa essere archiviato ovunque.

Questo può essere utile per archiviare o passare valori senza perdere il tipo e la struttura.

Per ottenere il valore dalla stringa serializzata, utilizzare la funzione unserialize(). La funzione **serialize()** gestisce tutti i tipi di variabili tranne il tipo resource. Possono essere elaborati da **serialize()** array che contengano riferimenti a se stessi. Saranno archiviati anche i riferimenti interni agli array e ai tipi object passati a **serialize()**

> **Nota:** Nella versione 3 di PHP si serializzano le proprietà dell'oggetto, ma non i metodi. Nella versione 4 viene superata questa limitazione e possono essere recuperati sia le proprietà sia i metodi. Per maggiori informazioni vedere la sezione Serializzare oggetti in Oggetti e Classi.

Esempio 1. Esempio di serialize()

```
// L'array multi-dimensionale $session_data contiene le informazioni della sessione
// per l'utente. Si userà serialize() per memorizzare le informazioni
// all'interno di un database alla fine della richiesta..

$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn,
      "UPDATE sessions SET data = ? WHERE id = ?");
$sqldata = array (serialize($session_data), $PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata)) {
    $stmt = odbc_prepare($conn,
     "INSERT INTO sessions (id, data) VALUES(?, ?)");
    if (!odbc_execute($stmt, &$sqldata)) {
    /* Qualcosa è andato storto */
    }
}
```

Vedere anche: unserialize().

settype (PHP 3, PHP 4 >= 4.0.0)

Definisce il tipo di una variabile

bool **settype** (mixed var, string type) \linebreak

Setta il tipo della variabile `var` a `type`.

Valori possibili di *type* sono:

- "boolean" (oppure, da PHP 4.2.0, "bool")
- "integer" (uppure, da PHP 4.2.0, "int")
- "float" (soltanto a partire dalla versione 4.2.0 di PHP, per le versioni precedenti usare "double", deprecated)
- "string"
- "array"
- "object"
- "null" (dalla versione PHP 4.0.8)

Restituisce `TRUE` se l'operazione ha successo; in caso contrario resituisce `FALSE`.

Esempio 1. Esempio di utilizzo di settype()

```
$foo = "5bar"; // string
$bar = true;   // boolean

settype($foo, "integer"); // $foo ora è 5   (integer)
settype($bar, "string");  // $bar ora è "1" (string)
```

Vedere anche gettype(), Casting del tipo e Manipolazione del tipo.

strval (PHP 3, PHP 4 >= 4.0.0)

Restituisce il valore di una variabile interpretato come stringa

string **strval** (mixed var) \linebreak

Restituisce il valore di *var* interpretato come stringa. Per maggiori informazioni sulla conversione a stringa, vedere la documentazione relativa alle variabili di tipo string.

var può essere una qualsiasi variabile scalare. Non è possibile usare **strval()** con array o oggetti.

Vedere anche floatval(), intval(), settype() e Manipolazione del tipo.

unserialize (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Crea un valore PHP a partire da una rappresentazione archiviata

mixed **unserialize** (string str) \linebreak

La funzione **unserialize()** prende il formato serializzato di una variabile (vedere serialize()) e la riporta a valore PHP. La funzione restituisce il valore ottenuto, che può essere di tipo integer, float, string, array oppure object.

> **Nota:** E' possibile impostare una funzione di callback che possa essere richiamata se, durante la fase di deserializzazione, occorre istanziare una classe indefinita. (per evitare di ottenere un tipo object incompleto "__PHP_Incomplete_Class".) Per definire il parametro 'unserialize_callback_func' si può agire sul `php.ini`, o usare ini_set() oppure con .htaccess-file Verrà utilizzata questa funzione ogni volta che occorre istanziare una classe indefinita. Per disabilitare questa opzione, lasciare vuoto questo settaggio.

Esempio 1. Esempio di unserialize_callback_func

```
$serialized_object='O:1:"a":1:{s:5:"value";s:3:"100";}';

ini_set('unserialize_callback_func','mycallback'); // imposta la funzione di callback

function mycallback($classname) {
    // semplicemente include un file contenente la definizione della classe
    // la variabile $classname indica di quale classe occorre la definizione
}
```

> **Nota:** Nella versione 3 di PHP, i metodi non erano preservati durante la fase di deserializzazione di un oggetto. In PHP 4 questo limite è stato superato e possono essere recuperati sia metodi sia proprietà. Per maggiori informazioni vedere la sezione Serializzare oggetti in Classi e oggetti

Esempio 2. Esempio di uso di unserialize()

```
// In quest esempio si usa unserialize() per caricare i dati di una sessione da un databa
// alla variabile $session_data. Questo esempio è complementare a quello illustrato
// nella funzione <function>serialize</function>.

$conn = odbc_connect ("webdb", "php", "chicken");
$stmt = odbc_prepare ($conn, "SELECT data FROM sessions WHERE id = ?");
$sqldata = array ($PHP_AUTH_USER);
if (!odbc_execute ($stmt, &$sqldata) || !odbc_fetch_into ($stmt, &$tmp)) {
    // se odbc_execute o odbc_fetch_into hanno un errore si predispone un ar-
ray vuoto
    $session_data = array();
} else {
    // ora abbiamo i dati serializzati in $tmp[0].
    $session_data = unserialize ($tmp[0]);
    if (!is_array ($session_data)) {
       // qualcosa è andato storto, si predispone un array vuoto
```

```
        $session_data = array();
    }
}
```

Vedere anche: serialize().

unset (unknown)

Cancella una data variabile

void **unset** (mixed var [, mixed var [, ...]]) \linebreak

> **Nota: unset()** è un costrutto del linguaggio.

unset() distrugge la variabile specificata. Occorre notare che in PHP 3 la funzione **unset()** restituiva sempre TRUE (in realtà era il valore 1). In PHP 4, invece, la funzione **unset()** non è più una vera funzione, ora è una istruzione. In questa veste non ritorna alcun valore, e cercare di ottenere un valore dalla funzione **unset()** produce un errore di parsing.

Esempio 1. Esempio di unset()

```
// Distrugge una singola variabile
unset ($foo);

// distrugge un singolo elemento di un array
unset ($bar['quux']);

// distrugge più di una variabile
unset ($foo1, $foo2, $foo3);
```

Il comportamento di **unset()** all'interno di una funzione può variare in funzione del tipo di variabile che si intende distruggere.

Se si applica **unset()** ad una variabile globale all'interno di una funzione, sarà distrutta solo la variabile locale. Nell'ambiente chiamante, la variabile manterrà il medesimo valore che aveva prima dell'uso di **unset()**.

```
function destroy_foo() {
    global $foo;
    unset($foo);
}
```

```
$foo = ’bar’;
destroy_foo();
echo $foo;
```

L’esempio precedente visualizzerà:

```
bar
```

Nel caso di una variabile PASSATA PER RIFERIMENTO ad una funzione, l’uso della funzione **unset()** distrugge solo la copia locale della variabile. Nell’ambiente chiamante, la variabile manterrà il medesimo valore che aveva prima dell’uso di **unset()**.

```
function foo(&$bar) {
    unset($bar);
    $bar = "blah";
}

$bar = ’something’;
echo "$bar\n";

foo($bar);
echo "$bar\n";
```

Il precedente esempio visualizzerà:

```
something
something
```

Se si usa **unset()** su una variabile statica all’interno di una funzione, **unset()** cancella la variabile e tutti i suoi riferimenti.

```
function foo() {
    static $a;
    $a++;
    echo "$a\n";

    unset($a);
}

foo();
```

```
foo();
foo();
```

L'esempio precedente visualizzerà:

```
1
2
3
```

Se si desidera cancellare una variabile globale dall'interno di una funzione, occorre usare **unset()** sull'array `$GLOBALS` nel seguente modo:

```
function foo() {
    unset($GLOBALS['bar']);
}

$bar = "something";
foo();
```

Vedere anche isset() e empty().

var_dump (PHP 3>= 3.0.5, PHP 4 >= 4.0.0)

Stampa delle informazioni relative ad una variabile

void **var_dump** (mixed expression [, mixed expression [, ...]]) \linebreak

Questa funzione restituisce delle informazioni strutturate riguardanti una espressione, tra cui il suo tipo e il suo valore. Gli array sono attraversati in maniera ricorsiva e i valori indentati per evidenziare la struttura.

> **Suggerimento:** Come con qualsiasi cosa che invia il risultato direttamente al browser, è possibile utilizzare la funzione output-control per catturare l'uscita di questa funzione e salvarla - per esempio - in una stringa.

Si confronti **var_dump()** con print_r().

```
<pre>
<?php
```

```
$a = array (1, 2, array ("a", "b", "c"));
var_dump ($a);

/* output:
array(3) {
  [0]=>
  int(1)
  [1]=>
  int(2)
  [2]=>
  array(3) {
    [0]=>
    string(1) "a"
    [1]=>
    string(1) "b"
    [2]=>
    string(1) "c"
  }
}

*/

$b = 3.1;
$c = TRUE;
var_dump($b,$c);

/* output:
float(3.1)
bool(true)

*/
?>
</pre>
```

var_export (PHP 4 >= 4.2.0)

Visualizza o restituisce una variabile in formato stringa

mixed **var_export** (mixed expression [, bool return]) \linebreak

Questa funzione restituisce informazioni strutturate sulla variabile che viene passata. Il comportamento è simile a var_dump() con la sola differenza che il valore restituito è codice PHP.

Si può anche ottenere la rappresentazione della variabile usando `TRUE` come secondo parametro della funzione.

Confrontare **var_export()** con var_dump().

```
<pre>
<?php
```

```
$a = array (1, 2, array ("a", "b", "c"));
var_export ($a);

/* output:
array (
  0 => 1,
  1 => 2,
  2 =>
  array (
    0 => 'a',
    1 => 'b',
    2 => 'c',
  ),
)
*/

$b = 3.1;
$v = var_export($b, TRUE);
echo $v;

/* output:
3.1
*/
?>
</pre>
```

C. Funzioni vpopmail

vpopmail_add_alias_domain (PHP 4 >= 4.0.5)

Aggiunge un alias per un dominio virtuale

bool **vpopmail_add_alias_domain** (string domain, string aliasdomain) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_add_alias_domain_ex (PHP 4 >= 4.0.5)

Aggiunge un alias ad un esistente dominio virtuale

bool **vpopmail_add_alias_domain_ex** (string olddomain, string newdomain) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_add_domain (PHP 4 >= 4.0.5)

Aggiunge un nuovo dominio virtuale

bool **vpopmail_add_domain** (string domain, string dir, int uid, int gid) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_add_domain_ex (PHP 4 >= 4.0.5)

Aggiunge un nuovo dominio virtuale

bool **vpopmail_add_domain_ex** (string domain, string passwd [, string quota [, string bounce [, bool apop]]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_add_user (PHP 4 >= 4.0.5)

Aggiunge un nuovo utente al dominio virtuale specificato

bool **vpopmail_add_user** (string user, string domain, string password [, string gecos [, bool apop]]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_alias_add (PHP 4 >= 4.1.0)

Inserisce un alias virtuale

bool **vpopmail_alias_add** (string user, string domain, string alias) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_alias_del (PHP 4 >= 4.1.0)

Elimina tutti gli alias virtuali di un utente

bool **vpopmail_alias_del** (string user, string domain) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_alias_del_domain (PHP 4 >= 4.1.0)

Elimina tutti gli alias virtuali di un dominio

bool **vpopmail_alias_del_domain** (string domain) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_alias_get (PHP 4 >= 4.1.0)

Riceve tutte le linee di un alias per un dominio

array **vpopmail_alias_get** (string alias, string domain) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_alias_get_all (PHP 4 >= 4.1.0)

Riceve tutte le linee di un alias per un dominio

array **vpopmail_alias_get_all** (string domain) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_auth_user (PHP 4 >= 4.0.5)

Tenta di validare un nome utente/dominio/password. Restituisce true/false

bool **vpopmail_auth_user** (string user, string domain, string password [, string apop]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_del_domain (PHP 4 >= 4.0.5)

Elimina un dominio virtuale

bool **vpopmail_del_domain** (string domain) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_del_domain_ex (PHP 4 >= 4.0.5)

Elimina un dominio virtuale

bool **vpopmail_del_domain_ex** (string domain) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_del_user (PHP 4 >= 4.0.5)

Elimina un utente da un dominio virtuale

bool **vpopmail_del_user** (string user, string domain) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_error (PHP 4 >= 4.0.5)

Riceve il messaggio testuale dell'ultimo errore di vpopmail. Restituisce string

string **vpopmail_error** (void) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_passwd (PHP 4 >= 4.0.5)

Cambia la password virtuale ad un utente

bool **vpopmail_passwd** (string user, string domain, string password) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

vpopmail_set_user_quota (PHP 4 >= 4.0.5)

Imposta la quota ad un utente virtuale

bool **vpopmail_set_user_quota** (string user, string domain, string quota) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

CI. Funzioni W32api

w32api_deftype (PHP 4 >= 4.2.0)

...) Definisce un tipo per l'uso con altre w32api_functions

int **w32api_deftype** (string typename, string member1_type, string member1_name) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

w32api_init_dtype (PHP 4 >= 4.2.0)

Crea un'istanza ai tipi di dati typename e li riempie con i valori val1, val2, la funzione quindi restituisce un DYNAPARM che può essere passato all'invocazione di una funzione come parametro

resource **w32api_init_dtype** (string typename, mixed val1, mixed val2) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

w32api_invoke_function (unknown)

....) Invoca la funzione funcname con gli argomenti passati dopo il nome della funzione

mixed **w32api_invoke_function** (string funcname) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

w32api_register_function (PHP 4 >= 4.2.0)

Registra la funzione function_name dalla libreria con PHP

bool **w32api_register_function** (string libary, string function_name) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

w32api_set_call_method (PHP 4 >= 4.2.0)

Imposta il metodo di chiamata usato

void **w32api_set_call_method** (int method) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

CII. WDDX Functions

These functions are intended for work with WDDX (http://www.openwddx.org/).

In order to use WDDX, you will need to install the expat library (which comes with apache 1.3.7 or higher) and recompile PHP with `--with-xml` and `--enable-wddx`.

> **Nota:** If you want to serialize non-ASCII characters you have to set the appropriate locale before doing so (see setlocale()).

All the functions that serialize variables use the first element of an array to determine whether the array is to be serialized into an array or structure. If the first element has string key, then it is serialized into a structure, otherwise, into an array.

Esempio 1. Serializing a single value

```
<?php
print wddx_serialize_value("PHP to WDDX packet example", "PHP packet");
?>
```

This example will produce:

```
<wddxPacket version='1.0'><header comment='PHP packet'/><data>
<string>PHP to WDDX packet example</string></data></wddxPacket>
```

Esempio 2. Using incremental packets

```
<?php
$pi = 3.1415926;
$packet_id = wddx_packet_start("PHP");
wddx_add_vars($packet_id, "pi");

/* Suppose $cities came from database */
$cities = array("Austin", "Novato", "Seattle");
wddx_add_vars($packet_id, "cities");

$packet = wddx_packet_end($packet_id);
print $packet;
?>
```

This example will produce:

```
<wddxPacket version='1.0'><header comment='PHP'/><data><struct>
<var name='pi'><number>3.1415926</number></var><var name='cities'>
<array length='3'><string>Austin</string><string>Novato</string>
<string>Seattle</string></array></var></struct></data></wddxPacket>
```

wddx_add_vars (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Add variables to a WDDX packet with the specified ID

wddx_add_vars (int packet_id, mixed name_var [, mixed ...]) \linebreak

wddx_add_vars() is used to serialize passed variables and add the result to the packet specified by the *packet_id*. The variables to be serialized are specified in exactly the same way as wddx_serialize_vars().

wddx_deserialize (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Deserializes a WDDX packet

mixed **wddx_deserialize** (string packet) \linebreak

wddx_deserialize() takes a *packet* string and deserializes it. It returns the result which can be string, number, or array. Note that structures are deserialized into associative arrays.

wddx_packet_end (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Ends a WDDX packet with the specified ID

string **wddx_packet_end** (int packet_id) \linebreak

wddx_packet_end() ends the WDDX packet specified by the *packet_id* and returns the string with the packet.

wddx_packet_start (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Starts a new WDDX packet with structure inside it

int **wddx_packet_start** ([string comment]) \linebreak

Use **wddx_packet_start()** to start a new WDDX packet for incremental addition of variables. It takes an optional *comment* string and returns a packet ID for use in later functions. It automatically creates a structure definition inside the packet to contain the variables.

wddx_serialize_value (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Serialize a single value into a WDDX packet

string **wddx_serialize_value** (mixed var [, string comment]) \linebreak

wddx_serialize_value() is used to create a WDDX packet from a single given value. It takes the value contained in *var*, and an optional *comment* string that appears in the packet header, and returns the WDDX packet.

wddx_serialize_vars (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Serialize variables into a WDDX packet

string **wddx_serialize_vars** (mixed var_name [, mixed ...]) \linebreak

wddx_serialize_vars() is used to create a WDDX packet with a structure that contains the serialized representation of the passed variables.

wddx_serialize_vars() takes a variable number of arguments, each of which can be either a string naming a variable or an array containing strings naming the variables or another array, etc.

Esempio 1. wddx_serialize_vars() example

```
<?php
$a = 1;
$b = 5.5;
$c = array("blue", "orange", "violet");
$d = "colors";

$clvars = array("c", "d");
print wddx_serialize_vars("a", "b", $clvars);
?>
```

The above example will produce:

```
<wddxPacket version='1.0'><header/><data><struct><var name='a'><number>1</number></var>
<var name='b'><number>5.5</number></var><var name='c'><array length='3'>
<string>blue</string><string>orange</string><string>violet</string></array></var>
<var name='d'><string>colors</string></var></struct></data></wddxPacket>
```

CIII. Funzioni relative al parser XML

Introduzione

XML

XML (eXtensible Markup Language) è un formato utilizzato per l'interscambio di documenti strutturati sul Web. Questo è uno standard definito da The World Wide Web consortium (W3C). Maggiori informazioni su XML e le relative tecnologie possono essere reperite all'indirizzo http://www.w3.org/XML/.

Installazione

Questa estensione utilizza il modulo expat, che può essere reperito all'indirizzo http://www.jclark.com/xml/. Il Makefile fornito con expat, per default, non compila la libreria, occorre utilizzare le seguenti istruzioni per il comando make:

```
libexpat.a: $(OBJS)
    ar -rc $@ $(OBJS)
    ranlib $@
```

Il sorgente in formato package RPM può essere reperito all'indirizzo http://sourceforge.net/projects/expat/.

Si noti che se si usa Apache 1.3.7 o successive, la libreria expat è già presente. In questo caso configurare PHP utilizzando `--with-xml` (senza specificare percorsi addizionali) e automaticamente verrà utilizzata la libreria expat presente in Apache.

Sui sistemi Unix eseguire **configure** con l'opzione `--with-xml`. Il compilatore è in grado di trovare sul sistema la libreria expat installata. Se si compila il PHP come modulo per Apache, dalla versione 1.3.9 e successive, il PHP automaticamente utilizzerà la libreria expat presente in Apache. Può essere necessario configurare le variabili d'ambiente CPPFLAGS e LDFLAGS prima di eseguire **configure** se si ha installato expat in qualche posto particolare.

Compila del PHP. *Tada!* Questo è quanto.

Note su questa estensione

Questa estensione di PHP implementa il supporto per il modulo expat di James Clark in PHP. Questo strumento permette il parsing, ma non la validazione, di documenti XML. Il modulo supporta tre tipi di codifica di caratteri già presenti in PHP: `US-ASCII`, `ISO-8859-1` e `UTF-8`. La codifica `UTF-16`

non è supportata.

Questa estensione permette di creare un parser XML e di definire dei *gestori* per i differenti eventi di XML. Ciascun parser XML ha diversi parametri configurabili.

I gestori di eventi XML definiti sono:

Tabella 1. Gestori XML supportati

Funzione PHP per attivare il gestore	Descrizione dell'evento
xml_set_element_handler()	L'evento 'Elemento' viene attivato quando il parser XML incontra i tag di apertura e chiusura. Esistono gestori separati per i tag di apertura e di chiusura.
xml_set_character_data_handler()	Sono dati tutti i contenuti dei documenti XML che non siano dei markup, compresi gli spazi tra i tag. Si noti che il parser XML non aggiunge ne rimuove spazi, è compito dell'applicazione decidere se gli spazi siano significativi o meno.
xml_set_processing_instruction_handler()	Ai programmatori PHP dovrebbe già essere familiare le istruzioni di processo (PIs). <?php ?> è un istruzione di processo dove *php* viene definito "PI target". La gestione di questi è specifica dell'applicazione, tranne che tutti i PI targets che iniziano con "XML", questi sono riservati.
xml_set_default_handler()	Tutto ciò che non rientra negli altri gestori ricade nel gestore di default. In questo gestore si ha elementi come la dichiarazione dei tipo documento.
xml_set_unparsed_entity_decl_handler()	Questo gestore viene richiamato per la gestione di entità non analizzate (NDATA).
xml_set_notation_decl_handler()	Questo gestore viene richiamato per la dichiarazione di una notazione.
xml_set_external_entity_ref_handler()	Questo gestore viene richiamato quando il parser XML incontra un riferimento ad una entità esterna analizzata. Questa può essere, ad esempio, un riferimento ad un file o ad un URL. Vedere esempio di entità esterne per una dimostrazione.

Case Folding

Le funzioni di gestione degli elementi possono ottenere i nomi dei propri elementi *case-folded*. Lo standard XML definisce il case-folding come "un processo applicato ad una sequenza di caratteri, nel quale quelli identificati come non-maiuscoli sono sostituiti dai corrispettivi caratteri maiuscoli". In altre parole, in XML il termine case-folding indica, semplicemente, la conversione a lettere maiuscole.

Per default, i nomi di tutti gli elementi passati alle funzioni di gestione sono case-folded. Questo

comportamento può essere verificato e controllato nel parser XML rispettivamente con le funzioni xml_parser_get_option() e xml_parser_set_option().

Codici di errore

Vengono definite le seguenti costanti per i codici di errore XML (come restituito da xml_parse()):

XML_ERROR_NONE
XML_ERROR_NO_MEMORY
XML_ERROR_SYNTAX
XML_ERROR_NO_ELEMENTS
XML_ERROR_INVALID_TOKEN
XML_ERROR_UNCLOSED_TOKEN
XML_ERROR_PARTIAL_CHAR
XML_ERROR_TAG_MISMATCH
XML_ERROR_DUPLICATE_ATTRIBUTE
XML_ERROR_JUNK_AFTER_DOC_ELEMENT
XML_ERROR_PARAM_ENTITY_REF
XML_ERROR_UNDEFINED_ENTITY
XML_ERROR_RECURSIVE_ENTITY_REF
XML_ERROR_ASYNC_ENTITY
XML_ERROR_BAD_CHAR_REF
XML_ERROR_BINARY_ENTITY_REF
XML_ERROR_ATTRIBUTE_EXTERNAL_ENTITY_REF
XML_ERROR_MISPLACED_XML_PI
XML_ERROR_UNKNOWN_ENCODING
XML_ERROR_INCORRECT_ENCODING
XML_ERROR_UNCLOSED_CDATA_SECTION
XML_ERROR_EXTERNAL_ENTITY_HANDLING

Codifica dei caratteri

L'estensione XML di PHP supporta il set di caratteri Unicode (http://www.unicode.org/) tramite differenti *codifiche di caratteri*. Esistono due tipi di codifiche di caratteri, *source encoding* e *target encoding*. Nella rappresentazione interna dei documenti il PHP utilizza sempre la codifica `UTF-8`.

La codifica del sorgente viene eseguita quando un documento XML viene analizzato. Durante la creazione di un parser XML, si può specificare la codifica del sorgente (questa codifica non potrà essere variata in seguito nel corso della vita del parser XML). Le codifiche supportate sono `ISO-8859-1`, `US-ASCII` e `UTF-8`. Le prime due sono codifiche a singolo byte, ciò significa che ciascun carattere è rappresentato da un byte singolo, mentre la codifica `UTF-8` può rappresentare caratteri composti da un numero variabile di bit (fino a 21) usando da uno fino a quattro byte. La codifica del sorgente utilizzata per default dal PHP è `ISO-8859-1`.

La codifica per il destinatario viene eseguita quando il PHP passa i dati alle funzioni di gestione del XML. Quando viene creato un parser XML, la codifica per il destinatario viene posta uguale alla codifica del sorgente, ma ciò può essere variato. La codifica per il destinatario viene applicata ai caratteri dei dati, ai nomi delle tag e alle istruzioni di processamento.

Se il parser XML incontra caratteri esterni al range dei caratteri rappresentabili dalla codifica, restituirà un errore.

Se il PHP incontra nel documento analizzato dei caratteri che non è in grado di rappresentare con la codifica scelta per il destinatario, "degrada" il carattere problematico. Attaualmente ciò significa

sostuire il carattere in questione con un punto interrogativo.

Alcuni esempi

Di seguito verranno illustrati alcuni esempi di script PHP per il parsing di documenti XML.

Esempio della struttura degli elementi XML

Il primo esempio visualizza, con indentazione, la struttura degli elementi di apertura di un documento.

Esempio 1. Visualizza la struttura degli elementi XML

```
$file = "data.xml";
$depth = array();

function startElement($parser, $name, $attrs) {
    global $depth;
    for ($i = 0; $i < $depth[$parser]; $i++) {
        print "  ";
    }
    print "$name\n";
    $depth[$parser]++;
}

function endElement($parser, $name) {
    global $depth;
    $depth[$parser]--;
}

$xml_parser = xml_parser_create();
xml_set_element_handler($xml_parser, "startElement", "endElement");
if (!($fp = fopen($file, "r"))) {
    die("could not open XML input");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("XML error: %s at line %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

Esempio di mappatura dei tag XML

Esempio 2. Conversione da XML a HTML

Il seguente esempio converte i tag di un documento XML in tag HTML. Gli elementi non trovati nella matrice delle tag saranno ignorati. Ovviamente questo esempio funziona solo con uno specifico tipo di documento XML:

```
$file = "data.xml";
$map_array = array(
    "BOLD"     => "B",
    "EMPHASIS" => "I",
    "LITERAL"  => "TT"
);

function startElement($parser, $name, $attrs) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "<$htmltag>";
    }
}

function endElement($parser, $name) {
    global $map_array;
    if ($htmltag = $map_array[$name]) {
        print "</$htmltag>";
    }
}

function characterData($parser, $data) {
    print $data;
}

$xml_parser = xml_parser_create();
// Si utilizza il case-folding per essere certi di trovare le tag in $map_array
xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, true);
xml_set_element_handler($xml_parser, "startElement", "endElement");
xml_set_character_data_handler($xml_parser, "characterData");
if (!($fp = fopen($file, "r"))) {
    die("Non si riesce ad aprire il documento XML");
}

while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("Errore XML: %s alla linea %d",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
xml_parser_free($xml_parser);
```

Esempio di entità XML esterna

Questo esempio evidenzia il codice XML. Si illustrerà come utilizzare il riferimento ad entità esterne per includere ed analizzare altri documenti, sarà illustrato anche come processare le PIs, ed il modo per determinare l'affidabilità del codice contenuto dalle PIs.

I documenti XML che possono essere usati per questo esempio sono presenti dopo l'esempio (`xmltest.xml` e `xmltest2.xml`.)

Esempio 3. Esempio di entità esterna

```
<?php
$file = "xmltest.xml";

function trustedFile($file) {
    // si considera affidabili soltanto i file locali del proprio utente
    if (!eregi("^([a-z]+)://", $file)
        && fileowner($file) == getmyuid()) {
            return true;
    }
    return false;
}

function startElement($parser, $name, $attribs) {
    print "&lt;<font color=\"#0000cc\">$name</font>";
    if (sizeof($attribs)) {
        while (list($k, $v) = each($attribs)) {
            print " <font color=\"#009900\">$k</font>=\"<font
                   color=\"#990000\">$v</font>\"";
        }
    }
    print "&gt;";
}

function endElement($parser, $name) {
    print "&lt;/<font color=\"#0000cc\">$name</font>&gt;";
}

function characterData($parser, $data) {
    print "<b>$data</b>";
}

function PIHandler($parser, $target, $data) {
    switch (strtolower($target)) {
        case "php":
            global $parser_file;
            // Se il documento analizzato è "affidabile", si può dire che è sicura
            // l'esecuzione del codice PHP presente in esso. In caso contrario si visuali
```

```
            // il codice.
            if (trustedFile($parser_file[$parser])) {
                eval($data);
            } else {
                printf("Codice PHP inaffidabile: <i>%s</i>",
                        htmlspecialchars($data));
            }
            break;
    }
}

function defaultHandler($parser, $data) {
    if (substr($data, 0, 1) == "&" && substr($data, -1, 1) == ";") {
        printf('<font color="#aa00aa">%s</font>',
                htmlspecialchars($data));
    } else {
        printf('<font size="-1">%s</font>',
                htmlspecialchars($data));
    }
}

function externalEntityRefHandler($parser, $openEntityNames, $base, $systemId,
                                  $publicId) {
    if ($systemId) {
        if (!list($parser, $fp) = new_xml_parser($systemId)) {
            printf("Non si riesce ad aprire l'entità %s at %s\n", $openEntityNames,
                   $systemId);
            return false;
        }
        while ($data = fread($fp, 4096)) {
            if (!xml_parse($parser, $data, feof($fp))) {
                printf("XML error: %s at line %d while parsing entity %s\n",
                       xml_error_string(xml_get_error_code($parser)),
                       xml_get_current_line_number($parser), $openEntityNames);
                xml_parser_free($parser);
                return false;
            }
        }
        xml_parser_free($parser);
        return true;
    }
    return false;
}

function new_xml_parser($file) {
    global $parser_file;

    $xml_parser = xml_parser_create();
    xml_parser_set_option($xml_parser, XML_OPTION_CASE_FOLDING, 1);
    xml_set_element_handler($xml_parser, "startElement", "endElement");
    xml_set_character_data_handler($xml_parser, "characterData");
    xml_set_processing_instruction_handler($xml_parser, "PIHandler");
    xml_set_default_handler($xml_parser, "defaultHandler");
    xml_set_external_entity_ref_handler($xml_parser, "externalEntityRefHandler");

    if (!($fp = @fopen($file, "r"))) {
        return false;
```

```
    }
    if (!is_array($parser_file)) {
        settype($parser_file, "array");
    }
    $parser_file[$xml_parser] = $file;
    return array($xml_parser, $fp);
}

if (!(list($xml_parser, $fp) = new_xml_parser($file))) {
    die("Non si riesce ad aprire il documento XML");
}

print "<pre>";
while ($data = fread($fp, 4096)) {
    if (!xml_parse($xml_parser, $data, feof($fp))) {
        die(sprintf("Errore XML: %s alla linea %d\n",
                    xml_error_string(xml_get_error_code($xml_parser)),
                    xml_get_current_line_number($xml_parser)));
    }
}
print "</pre>";
print "parsing completato\n";
xml_parser_free($xml_parser);

?>
```

Esempio 4. xmltest.xml

```
<?xml version='1.0'?>
<!DOCTYPE chapter SYSTEM "/just/a/test.dtd" [
<!ENTITY plainEntity "FOO entity">
<!ENTITY systemEntity SYSTEM "xmltest2.xml">
]>
<chapter>
 <TITLE>Title &plainEntity;</TITLE>
 <para>
  <informaltable>
   <tgroup cols="3">
    <tbody>
     <row><entry>a1</entry><entry morerows="1">b1</entry><entry>c1</entry></row>
     <row><entry>a2</entry><entry>c2</entry></row>
     <row><entry>a3</entry><entry>b3</entry><entry>c3</entry></row>
    </tbody>
   </tgroup>
  </informaltable>
 </para>
 &systemEntity;
 <section id="about">
  <title>Circa questo documento</title>
  <para>
   <!-- questo è un commento -->
   <?php print 'Ciao!  Questa è la versione di PHP '.phpversion(); ?>
```

```
  </para>
 </section>
</chapter>
```

Questo file è stato richiamato da `xmltest.xml`:

Esempio 5. xmltest2.xml

```
<?xml version="1.0"?>
<!DOCTYPE foo [
<!ENTITY testEnt "Entità di test">
]>
<foo>
   <element attrib="value"/>
   &testEnt;
   <?php print "Questa è una ulteriore riga di codice PHP eseguito."; ?>
</foo>
```

utf8_decode (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Converte una stringa con caratteri ISO-8859-1 codificati con UTF-8 in formato ISO-8859-1 singolo byte.

string **utf8_decode** (string data) \linebreak

Questa funzione decodifica il parametro *data*, considerato codificato in `UTF-8`, alla codifica `ISO-8859-1`.

Vedere anche utf8_encode() per dettagli sulla codifica UTF-8.

utf8_encode (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Codifica una stringa da ISO-8859-1 a UTF-8

string **utf8_encode** (string data) \linebreak

Questa funzione converte la stringa *data* al formato `UTF-8`, e restituisce la nuova versione. `UTF-8` è il meccanismo standard utilizzato da Unicode per la codifica dei valori *wide character* in un flusso di byte. La codifica `UTF-8` è trasparente ai caratteri ASCII, è auto-sincronizzata (per un programma è possibile determinare dove iniziano i caratteri in un flusso di dati) e può essere usata nelle normali funzioni di confronto di stringhe per i sort e simili. Il PHP codifica i caratteri `UTF-8` fino a quattro byte come segue:

Tabella 1. Codifica UTF-8

bytes	bits	rappresentazione
1	7	0bbbbbbb
2	11	110bbbbb 10bbbbbb
3	16	1110bbbb 10bbbbbb 10bbbbbb
4	21	11110bbb 10bbbbbb 10bbbbbb 10bbbbbb

Ciascuna *b* rappresenta un bit che può essere utilizzato per registrare le informazioni del carattere.

xml_error_string (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

restituisce la stringa di errore dal parser XML

string **xml_error_string** (int codice_errore) \linebreak

codice_errore

Un codice di errore da xml_get_error_code().

La funzione restituisce una stringa con la descrizione del codice di errore *codice_errore*, oppure `FALSE` se non si trova una descrizione.

xml_get_current_byte_index (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce il corrente indice di posizione per un parser XML

int **xml_get_current_byte_index** (resource parser) \linebreak

parser

Il riferimento al parser XML da cui si vuole ottenere la posizione.

Questa funzione restituisce FALSE se *parser* non si riferisce ad un parser valido, altrimenti la funzione restituisce in quale byte si trova attualmente il parser nel buffer dei dati (partendo da 0).

xml_get_current_column_number (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce il numero di colonna corrente di un parser XML

int **xml_get_current_column_number** (resource parser) \linebreak

parser

Il riferimento al parser XML da cui ottenere il numero di colonna.

Questa funzione restituisce FALSE se *parser* non si riferisce ad un parser valido, altrimenti restituisce in quale colonna sulla linea corrente (come indicato da xml_get_current_line_number()) si trova il parser.

xml_get_current_line_number (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce il numero di linea corrente da un parser XML

int **xml_get_current_line_number** (resource parser) \linebreak

parser

Il riferimento al parser XML da cui si vuole ottenere il numero di linea.

Questa funzione restituisce FALSE se *parser* non si riferisce ad un parser valido, altrimenti la funzione restituisce in quale linea del buffer dati si trova attualmente il parser.

xml_get_error_code (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce il codice di errore dal parser XML

int **xml_get_error_code** (resource parser) \linebreak

parser

Il riferimento al parser XML da cui si vuole ottenere il codice di errore.

Questa funzione restituisce FALSE se *parser* non si riferisce ad un parser valido, altrimenti la funzione restituisce uno dei codici di errore elencati in codici di errore.

xml_parse (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Inizia il parsing di un documento XML

bool **xml_parse** (resource parser, string data [, bool e_finale]) \linebreak

parser

Il riferimento al parser XML da utilizzare.

data

Segmento di dati da analizzare. Un documento può essere elaborato a segmenti richiamando **xml_parse()** diverse volte con nuovi dati, con il parametro *e_finale* settato a TRUE quando si elabora l'ultimo segmento.

e_finale (optional)

Se valorizzato a TRUE, il parametro *data* rappresenta l'ultimo segmento dei dati inviati al parser.

Quando si esegue il parsing di un documento XML, i gestori configurati per gli eventi vengono richiamati tutte le volte che è necessario, dopo i quali questa funzione restituisce TRUE oppure FALSE.

La funzione restituisce TRUE se il parsing riesce, altrimenti restituisce FALSE se non riesce o se il parametro *parser* si riferisce ad un parser non valido. In caso di parsing non riuscito, si può recuperare informazioni sull'errore usando xml_get_error_code(), xml_error_string(), xml_get_current_line_number(), xml_get_current_column_number() e xml_get_current_byte_index().

xml_parse_into_struct (PHP 3>= 3.0.8, PHP 4 >= 4.0.0)

Analisi di dati XML in una struttura a matrice

int **xml_parse_into_struct** (resource parser, string data, array &valori, array &indice) \linebreak

Questa funzione registra un file XML in 2 strutture a matrice, una (*indice*) contiene i puntatori alla posizione dei valori nella matrice *valori*. Questi due parametri devono essere passati per riferimento.

Nell’esempio seguente si illustra la struttura interna delle matrici generata dalla funzione. Qui si userà una semplice struttura con il tag `note` inserito all’interno del tag `para`, e quindi si analizzerà questo visualizzando la struttura ottenuta:

```
$simple = "<para><note>simple note</note></para>";
$p = xml_parser_create();
xml_parse_into_struct($p,$simple,$vals,$index);
xml_parser_free($p);
echo "Array indice\n";
print_r($index);
echo "\narray dei valori\n";
print_r($vals);
```

Eseguendo questo codice si otterrà:

```
Array indice
Array
(
    [PARA] => Array
        (
            [0] => 0
            [1] => 2
        )

    [NOTE] => Array
        (
            [0] => 1
        )

)

Array dei valori
Array
(
    [0] => Array
        (
            [tag] => PARA
            [type] => open
            [level] => 1
        )

    [1] => Array
        (
            [tag] => NOTE
            [type] => complete
            [level] => 2
            [value] => simple note
        )

    [2] => Array
        (
            [tag] => PARA
```

```
            [type] => close
            [level] => 1
        )

)
```

Un parsing event-driven (basato sulla libreria expat) può essere molto complicato se si deve trattare un documento XML complesso. Questa funzione non produce un oggetto in stile DOM, ma genera una struttura che permette di essere gestita a modo di albero. Quindi si può facilmente creare oggetti rappresentanti i dati del file XML. Si consideri il seguente file XML rappresentante un piccolo database di informazioni sugli aminoacidi.

Esempio 1. moldb.xml - piccolo database di informazioni sulle molecole

```
<?xml version="1.0"?>
<moldb>

    <molecule>
        <name>Alanine</name>
        <symbol>ala</symbol>
        <code>A</code>
        <type>hydrophobic</type>
    </molecule>

    <molecule>
        <name>Lysine</name>
        <symbol>lys</symbol>
        <code>K</code>
        <type>charged</type>
    </molecule>

</moldb>
```

Un codice per analizzare il documento e generare gli oggetti appropriati:

Esempio 2. parsemoldb.php - analizza moldb.xml e genera una matrice di oggetti molecole

```
<?php

class AminoAcid {
    var $name;  // aa nome
    var $symbol;    // simbolo di tre lettere
    var $code;  // codice di una lettera
    var $type;  // hydrophobic, charged or neutral

    function AminoAcid ($aa) {
        foreach ($aa as $k=>$v)
            $this->$k = $aa[$k];
```

```
    }
}

function readDatabase($filename) {
    // legge il database degli aminoacidi
    $data = implode("",file($filename));
    $parser = xml_parser_create();
    xml_parser_set_option($parser,XML_OPTION_CASE_FOLDING,0);
    xml_parser_set_option($parser,XML_OPTION_SKIP_WHITE,1);
    xml_parse_into_struct($parser,$data,$values,$tags);
    xml_parser_free($parser);

    // loop attraverso la struttura
    foreach ($tags as $key=>$val) {
        if ($key == "molecule") {
            $molranges = $val;
            // ciascuna coppia di elementi della matrice contigui
            // sono il limite inferiore e superiore per la definizione di cias-
cuna molecola
            for ($i=0; $i < count($molranges); $i+=2) {
                    $offset = $molranges[$i] + 1;
                $len = $molranges[$i + 1] - $offset;
                $tdb[] = parseMol(array_slice($values, $offset, $len));
            }
        } else {
            continue;
        }
    }
    return $tdb;
}

function parseMol($mvalues) {
    for ($i=0; $i < count($mvalues); $i++)
        $mol[$mvalues[$i]["tag"]] = $mvalues[$i]["value"];
    return new AminoAcid($mol);
}

$db = readDatabase("moldb.xml");
echo "** Database di oggetti Aminoacidi:\n";
print_r($db);

?>
```

Dopo l’esecuzione di `parsemoldb.php`, la variabile `$db` contiene una matrice di oggetti `AminoAcid`, e ciò viene confermato dall’output dello script:

```
** Database di oggetti Aminoacidi:
Array
(
    [0] => aminoacid Object
        (
            [name] => Alanine
            [symbol] => ala
```

```
                [code] => A
                [type] => hydrophobic
            )

        [1] => aminoacid Object
            (
                [name] => Lysine
                [symbol] => lys
                [code] => K
                [type] => charged
            )

)
```

xml_parser_create (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Crea un parser XML

resource **xml_parser_create** ([string codifica]) \linebreak

codifica (optional)

Quale codifica di carattere utilizzare. Sono supportate le seguenti codifiche di carattere:

`ISO-8859-1` (default)

`US-ASCII`

`UTF-8`

Questa funzione crea un parser XML e restituisce un handle da usarsi con le altre funzioni XML. Restituisce `FALSE` in caso di errore.

xml_parser_create_ns (PHP 4 >= 4.0.5)

Crea un parser XML

resource **xml_parser_create_ns** ([string encoding [, string sep]]) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xml_parser_free (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Cancella un parser XML

bool **xml_parser_free** (resource parser) \linebreak

parser

Riferimento al parser XML da cancellare.

Questa funzione restituisce FALSE se *parser* non si riferisce ad un parser valido, altrimenti cancella il parser e restituisce TRUE

xml_parser_get_option (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Restituisce le opzioni da un parser XML

mixed **xml_parser_get_option** (resource parser, int opzione) \linebreak

parser

Riferimento al parser XML da cui ottenere l'opzione

opzione

Quale opzione ottenere. Vedere xml_parser_set_option() per l'elenco delle opzioni.

Questa funzione restituisce FALSE se *parser* non si riferisce ad un parser valido, o se l'opzione non può essere valorizzata. Altrimenti viene restituito il valore dell'opzione.

Vedere xml_parser_set_option() per l'elenco delle opzioni.

xml_parser_set_option (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Valorizza un'opzione di un parser XML

bool **xml_parser_set_option** (resource parser, int opzione, mixed valore) \linebreak

parser

Riferimento al parser XML di cui si vuole valorizzare l'opzione,

opzione

Quale opzione valorizzare. Vedere più avanti.

valore

Nuovo valore dell'opzione.

Questa funzione restituisce FALSE se *parser* non si riferisce ad un parser valido, o se l'opzione non può essere valorizzata. Altrimenti l'opzione viene valorizzata e la funzione restituisce TRUE.

Sono disponibili le seguenti opzioni:

Tabella 1. Opzioni del parser XML

Costante dell'opzione	tipo di dato	Descrizione
XML_OPTION_CASE_FOLDING	integer	Controlla se il case-folding è abilitato per questo parser XML. Abilitato per default.
XML_OPTION_TARGET_ENCODING	string	Indica quale target encoding utilizzare in questo parser XML. Per default, è valorizzato con la medesima codifica del sorgente utilizzata da xml_parser_create(). Le codifiche supportate per il destinatario sono ISO-8859-1, US-ASCII e UTF-8.

xml_set_character_data_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Valorizza il gestore dei dati

bool **xml_set_character_data_handler** (resource parser, string gestore) \linebreak

Valorizza la funzione di gestione dei dati per il parser XML *parser*. Il parametro *gestore* è una stringa contenente il nome di una funzione che deve esistere quando viene richiamata la funzione xml_parse() per il *parser*.

La fuzione indicata in *gestore* deve accettare due parametri: ***gestore*** (resource parser, string dati) \linebreak

parser

Il primo parametro, *parser*, è un riferimento al parser XML chiamante il gestore.

dati

Il secondo parametro, *dati*, è una stringa contenente i dati.

Se il nome della funzione di gestione viene indicato come una stringa vuota o FALSE il gestore in questione viene disabilitato.

La funzione restituisce TRUE se il gestore viene valorizzato, oppure FALSE se *parser* non indica un parser.

Nota: Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

xml_set_default_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Valorizza il gestore di default

bool **xml_set_default_handler** (resource parser, string gestore) \linebreak

Questa funzione valorizza il gestore di default per *parser*. Il parametro *gestore* è una stringa contenente il nome di una funzione che deve esistere quando viene eseguito xml_parse() per il *parser* XML.

La funzione indicata da *gestore* deve accettare due parametri: ***gestore*** (resource parser, string dati) \linebreak

parser

Il primo parametro, *parser*, è un riferimento al parser XML chiamante il gestore.

dati

Il secondo parametro, *dati*, contiene i dati. Questi possono essere dichiarazioni XML, dichiarazioni di tipo documento, entità oppure altri dati per i quali non è definito il gestore.

Se il nome della funzione del gestore viene valorizzato con una stringa vuota oppure a FALSE, il gestore in questione viene disabilitato.

La funzione restituisce TRUE se il gestore viene attivato, FALSE se *parser* non indica un parser XML.

Nota: Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

xml_set_element_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Valorizza i gestori di inizio e fine elemento

bool **xml_set_element_handler** (resource parser, string gestore_inizio_elemento, string gestore_fine_elemento) \linebreak

La funzione indica le funzioni di gestione di inizio e fine elemento per il *parser* XML. *gestore_inizio_elemento* e *gestore_fine_elemento* sono stringhe contenenti i nomi di funzioni che devono esistere quando viene eseguito xml_parse() per il *parser*.

La funzione indicata da *gestore_inizio_elemento* deve accettare tre parametri: ***gestore*** (resource parser, string nome, array attibuti) \linebreak

parser

Il primo parametro, *parser*, è il riferimento al parser XML chiamante il gestore.

nome

Il secondo parametro, *nome*, contiene il nome dell'elemento per il quale viene chiamato il gestore. Se è attivo il case-folding per questo parser, il nome dell'elemento sarà in maiuscolo.

attributi

Il terzo parametro, *attributi*, contiene un array associativo con gli attributi dell'elemento (se presenti). Le chiavi di questo array sono i nomi degli attributi, mentre i valori delle chiavi sono i valori degli attributi. I nomi degli attributi sono case-folded allo stesso modo dei nomi degli elementi. I valori degli attributi *non* lo sono.

L'ordine originale degli attributi può essere recuperato attraversando *attributi* in modo normale utilizzando la funzione each(). La prima chiave dell'array è il primo attributo, e così via.

La funzione indicata da *gestore_fine_elemento* deve accettare due parametri: ***gestore*** (resource parser, string nome) \linebreak

parser

Il primo parametro, *parser*, è il riferimento al parser XML chiamante il gestore.

nome

Il secondo parametro, *nome*, contiene il nome dell'elemento per il quale viene chiamato il gestore. Se è attivo il case-folding per questo parser, il nome dell'elemento sarà in maiuscolo.

Se il nome della funzione del gestore viene valorizzato con una stringa vuota oppure a FALSE, il gestore in questione viene disabilitato.

La funzione restituisce TRUE se il gestore viene attivato, FALSE se *parser* non indica un parser XML.

> **Nota:** Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

xml_set_end_namespace_decl_handler (PHP 4 >= 4.0.5)

Valorizza il gestore dei dati

bool **xml_set_end_namespace_decl_handler** (resource pind, string hdl) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

Nota: Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

xml_set_external_entity_ref_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Valorizza il gestore dei riferimenti a entità esterne

bool **xml_set_external_entity_ref_handler** (resource parser, string gestore) \linebreak

Indica al parser XML *parser* la funzione per la gestione dei riferimenti a entità esterne. Il *gestore* è una stringa contenente il nome di una funzione che deve esistere quando viene eseguita la funzione xml_parse() per il *parser*.

La funzione indicata da *gestore* deve accettare cinque parametri, e dovrebbe restituire un intero. Se il valore restituito dal gestore è FALSE (valore assunto per default in caso di nessun valore restituito), il parser XML ferma l'elaborazione e xml_get_error_code() restituirà XML_ERROR_EXTERNAL_ENTITY_HANDLING. ***gestore*** (resource parser, string nomi_entita_aperte, string base, string system_id, string public_id) \linebreak

parser

Il primo parametro, *parser*, è il riferimento al parser XML chiamante il gestore.

nome_entita_aperte

Il secondo parametro, *nomi_entita_aperte*, è una lista di nomi, separati da spazio, delle entità che saranno aperte per il parsing di questa entità (compreso il nome dell'entità indicata)

base

Questa è la base per la risoluzione dell'identificatore system (*systemid*) delle entità esterne. Attualmente questo parametro è sempre valorizzato con una stringa vuota.

system_id

Il quarto parametro, *system_id*, è l'identificatore system come specificato nella dichiarazione dell'entità.

public_id

Il quinto parametro, *public_id*, è l'identificatore public come specificato nella dichiarazione dell'entità, oppure una stringa vuota se non viene specificato; lo spazio nell'identificatore public è normalizzato come richiesto dalle specifiche XML.

Se il nome della funzione del gestore viene valorizzato con una stringa vuota oppure a FALSE, il gestore in questione viene disabilitato.

La funzione restituisce TRUE se il gestore viene attivato, FALSE se *parser* non indica un parser XML.

> **Nota:** Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

xml_set_notation_decl_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Valorizza il gestore delle dichiarazione delle notazioni

bool **xml_set_notation_decl_handler** (resource parser, string handler) \linebreak

Indica al parser XML *parser* la funzione per la gestione delle dichiarazioni delle notazioni. Il *gestore* è una stringa contenente il nome di una funzione che deve esistere quando viene eseguita la funzione xml_parse() per il *parser*.

La dichiarazione di una notazione è una parte della DTD del documento ed ha il seguente formato:

```
<!NOTATION
     name {system_id |
     public_id}>
```

Vedere la sezione 4.7 delle specifiche XML 1.0 (http://www.w3.org/TR/1998/REC-xml-19980210#Notations) per la definizione delle dichiarazioni delle notazioni.

La funzione indicata da *gestore* deve accettare cinque parametri: ***gestore*** (resource parser, string nome_notazione, string base, string system_id, string public_id) \linebreak

parser

Il primo parametro, *parser*, è il riferimento al parser XML chiamante il gestore.

nome_notazione

Questo è il parametro *name* della notazione, come dal formato descritto in precedenza.

base

Questa è la base per la risoluzione dell'identificatore system (*system_id*) delle entità esterne. Attualmente questo parametro è sempre valorizzato con una stringa vuota.

system_id

Identificatore system della dichiarazione della notazione esterna.

public_id

Identificatore public della dichiarazione della notazione esterna.

Se il nome della funzione del gestore viene valorizzato con una stringa vuota oppure a `FALSE`, il gestore in questione viene disabilitato.

La funzione restituisce `TRUE` se il gestore viene attivato, `FALSE` se *`parser`* non indica un parser XML.

> **Nota:** Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

xml_set_object (PHP 4 >= 4.0.0)

Utilizza il parser XML all'interno di un oggetto

void **xml_set_object** (resource parser, object &oggetto) \linebreak

Questa funzione permette l'uso del *`parser`* all'interno di un *`oggetto`*. Tutte le funzioni di callback possono essere indicate con xml_set_element_handler() e simili, e sono assunte come metodi dell'*`oggetto`*.

```
<?php
class xml  {
    var $parser;

    function xml()
    {
        $this->parser = xml_parser_create();

        xml_set_object($this->parser, &$this);
        xml_set_element_handler($this->parser, "tag_open", "tag_close");
        xml_set_character_data_handler($this->parser, "cdata");
    }

    function parse($data)
    {
        xml_parse($this->parser, $data);
    }

    function tag_open($parser, $tag, $attributes)
    {
        var_dump($parser, $tag, $attributes);
    }

    function cdata($parser, $cdata)
    {
        var_dump($parser, $cdata);
    }

    function tag_close($parser, $tag)
    {
        var_dump($parser, $tag);
    }
```

```
} // fine della classe xml

$xml_parser = new xml();
$xml_parser->parse("<A ID='hallo'>PHP</A>");
?>
```

xml_set_processing_instruction_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Indica il gestore delle istruzioni di processo (PI)

bool **xml_set_processing_instruction_handler** (resource parser, string gestore) \linebreak

Indica al parser XML *parser* la funzione per la gestione delle istruzioni di processo (PI). Il *gestore* è una stringa contenente il nome di una funzione che deve esistere quando viene eseguita la funzione xml_parse() per il *parser*.

Le istruzioni di processo hanno il seguente formato:

```
<?
       target
       data?>
```

Si può inserire codice PHP all'interno di questo tipo di tag, ma occorre fare attenzione ad una limitazione: in una PI XML, il tag di fine PI (`?>`) non può essere tra apici, pertanto questa seguenza di caratteri non dovrebbe apparire nel codice PHP che si inserisce nel documento XML con le PIs. Se ciò accade il resto del codice PHP, come il "reale" tag di fine PI, saranno trattati come caratteri di dati.

La funzione indicata da *gestore* deve accettare tre parametri: ***gestore*** (resource parser, string target, string dati) \linebreak

parser

Il primo parametro, *parser*, è il riferimento al parser XML chiamante il gestore.

target

Il secondo parametro, *target*, contiene la PI target.

dati

Il terzo parametro, *dati*, contiene i dati del PI.

Se il nome della funzione del gestore viene valorizzato con una stringa vuota oppure a `FALSE`, il gestore in questione viene disabilitato.

La funzione restituisce `TRUE` se il gestore viene attivato, `FALSE` se *parser* non indica un parser XML.

Nota: Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

xml_set_start_namespace_decl_handler (PHP 4 >= 4.0.5)

Valorizza il gestore dei caratteri di dati

bool **xml_set_start_namespace_decl_handler** (resource pind, string hdl) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

Nota: Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

xml_set_unparsed_entity_decl_handler (PHP 3>= 3.0.6, PHP 4 >= 4.0.0)

Valorizza il gestore delle dichiarazioni di entità non analizzate

bool **xml_set_unparsed_entity_decl_handler** (resource parser, string gestore) \linebreak

Indica al parser XML `parser` la funzione per la gestione delle dichiarazioni di entità non analizzate. Il `gestore` è una stringa contenente il nome di una funzione che deve esistere quando viene eseguita la funzione xml_parse() per il `parser`.

Questo gestore viene richiamato quando il parser XML incontra una dichiarazione di entità esterna con una dichiarazione NDATA, tipo la seguente:

```
<!ENTITY name {publicId | systemId}
        NDATA notationName>
```

Vedere la sezione 4.2.2 delle specifiche di XML 1.0 (http://www.w3.org/TR/1998/REC-xml-19980210#sec-external-ent) per la definizione di notazioni dichiarate in entità esterne.

La funzione indicata da `gestore` deve accettare sei parametri: **`gestore`** (resource parser, string nome_entità, string base, string system_id, string public_id, string nome_notazione) \linebreak

`parser`

Il primo parametro, `parser`, è il riferimento al parser XML chiamante il gestore.

nome_entità

Il nome dell'entità che sta per essere definita.

base

Questa è la base per la risoluzione dell'identificatore system (*systemid*) delle entità esterne. Attualmente questo parametro è sempre valorizzato con una stringa vuota.

system_id

Identificatore system per l'entità esterna.

public_id

Identificatore public per l'entità esterna.

nome_notazione

Nome della notazione di questa entità (vedere xml_set_notation_decl_handler()).

Se il nome della funzione del gestore viene valorizzato con una stringa vuota oppure a FALSE, il gestore in questione viene disabilitato.

La funzione restituisce TRUE se il gestore viene attivato, FALSE se *parser* non indica un parser XML.

Nota: Invece di un nome di funzione è possibile passare un vettore contenente un riferimento ad un oggetto e un nome di metodo.

CIV. Funzioni XMLRPC

Attenzione

Questo modulo è *SPERIMENTALE*. Ovvero, il comportamento di queste funzioni, i nomi di queste funzioni, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questo modulo è a vostro rischio.

xmlrpc_decode (PHP 4 >= 4.1.0)

Decodifica XML nei nativi tipi di PHP

array **xmlrpc_decode** (string xml [, string encoding]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_decode_request (PHP 4 >= 4.1.0)

Decodifica XML nei nativi tipi di PHP

array **xmlrpc_decode_request** (string xml, string method [, string encoding]) \linebreak

> **Attenzione**
>
> Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_encode (PHP 4 >= 4.1.0)

Genera XML per un valore PHP

string **xmlrpc_encode** (mixed value) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_encode_request (PHP 4 >= 4.1.0)

Genera XML per un metodo di richiesta

string **xmlrpc_encode_request** (string method, mixed params) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_get_type (PHP 4 >= 4.1.0)

Riceve il tipo di xmlrpc per un valore PHP. Maggiormente ì usato per le stringhe base64 e datetime

string **xmlrpc_get_type** (mixed value) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_parse_method_descriptions (PHP 4 >= 4.1.0)

Decodifica XML in una lista di method descriptions

array **xmlrpc_parse_method_descriptions** (string xml) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_server_add_introspection_data (PHP 4 >= 4.1.0)

Aggiunge documentazione introspettiva

int **xmlrpc_server_add_introspection_data** (resource server, array desc) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_server_call_method (PHP 4 >= 4.1.0)

Struttura le richieste XML e i metodi di chiamata

mixed **xmlrpc_server_call_method** (resource server, string xml, mixed user_data [, array output_options]) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_server_create (PHP 4 >= 4.1.0)

Crea un server xmlrpc

resource **xmlrpc_server_create** (void) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_server_destroy (PHP 4 >= 4.1.0)

Distrugge le risorse del server

void **xmlrpc_server_destroy** (resource server) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_server_register_introspection_callback (PHP 4 >= 4.1.0)

Registra una funzione PHP per generare la documentazione

bool **xmlrpc_server_register_introspection_callback** (resource server, string function) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_server_register_method (PHP 4 >= 4.1.0)

Registra una funzione PHP nel metodo di risorsa abbinato method_name

bool **xmlrpc_server_register_method** (resource server, string method_name, string function) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xmlrpc_set_type (PHP 4 >= 4.1.0)

Imposta il tipo di xmlrpc, base64 o datetime, per un valore di stringa PHP

bool **xmlrpc_set_type** (string value, string type) \linebreak

Attenzione

Questa funzione è *SPERIMENTALE*. Ovvero, il comportamento di questa funzione, il nome di questa funzione, in definitiva tutto ciò che è documentato qui può cambiare nei futuri rilasci del PHP senza preavviso. Siete avvisati, l'uso di questa funzione è a vostro rischio.

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

CV. Funzioni XSLT

Introduzione

XSLT e Sablotron

XSLT (Extensible Stylesheet Language (XSL) Transformations) è un linguaggio per trasformare documenti XML in altri documenti XML. E' uno standard definito dal World Wide Web consortium (W3C). Informazioni circa l' XSLT e le relative tecnologie possono essere trovate su http://www.w3.org/TR/xslt.

Installazione

Questa estensione usa Sablotron e expat, i quali possono essere entrambi trovati su http://www.gingerall.com/. I binari sono forniti assieme al codice sorgente.

Su UNIX, fai partire **configure** con l'opzione `--enable-xslt --with-xslt-sablot`. La libreria Sablotron dovrebbe essere installata da qualche parte dove il compilatore può trovarla.

Questa estensione

L'estensione PHP fornisce un processore indipendente API alle trasformazioni XSLT. Attualmente questa estensione supporta solo la libreria Sablotron della Ginger Alliance. Il supporto è pianificato per le altre librerie, come la libreria Xalan o la libreria libxslt.

> **Nota:** Questa estensione è differente dall'estensione sablotron distribuita con le versioni del PHP precedenti PHP 4.1, attualmente è supportata nel PHP 4.1 solo la nuova estensione XSLT. Se hai bisogno di supporto per le vecchie estensioni, fai la tua domanda sulla mailing list php-general@lists.php.net.

xslt_create (PHP 4 >= 4.0.3)

Crea un nuovo processore XSLT

resource **xslt_create** (void) \linebreak

Crea e restituisce un nuovo processore XSLT risorsa per la modifica dalle altre funzioni XSLT.

xslt_errno (PHP 4 >= 4.0.3)

Restituisce un numero di errore

int **xslt_errno** (resource xh) \linebreak

Restituisce un codice di errore che descrive l'ultimo errore successo sul processore XSLT indicato.

xslt_error (PHP 4 >= 4.0.3)

Restituisce una stringa di errore

mixed **xslt_error** (resource xh) \linebreak

Restituisce una stringa che descrive l'ultimo errore restituito dal processore XSLT indicato.

Esempio 1. Gli errori di Handling usano le funzioni xslt_error() e xslt_errno().

```
<?php

$xh = xslt_create();
$result = xslt_process($xh, 'dog.xml', 'pets.xsl');
if (!$result) {
    die(sprintf("Cannot process XSLT document [%d]: %s",
                xslt_errno($xh), xslt_error($xh)));
}

print($result);

xslt_free($xh);
?>
```

xslt_free (PHP 4 >= 4.0.3)

Libera un processore XSLT

void **xslt_free** (resource xh) \linebreak

Libera il processore XSLT identificato dall'handle dato.

xslt_process (PHP 4 >= 4.0.3)

Esegue una trasformazione XSLT

mixed **xslt_process** (resource xh, string xml, string xsl [, string result [, array arguments [, array parameters]]]) \linebreak

La funzione xslt_process() è una delle più importanti della nuova estensione XSLT. Permette di eseguire una trasformazione XSLT usando quasi ogni tipo di fonte per l'input. Questa è un completamento mediante l'uso dell'argomento buffers -- un concetto preso dal processore XSLT Sablotron (attualmente l'unico processore XSLT supportato da questa estensione).

Il più semplice tipo di trasformazione con la funzione **xslt_process()** è la trasformazione di un file XML con un file XSLT, mettendo il risultato in un terzo file contenente un nuovo documento XML (o HTML). Fare questo con Sablotron è davvero molto semplice...

Esempio 1. Uso di xslt_process() per trasformare un file XML e un file XSL in un nuovo file XML

```
<?php

// Allocate a new XSLT processor
$xh = xslt_create();

// Process the document
if (xslt_process($xh, 'sample.xml', 'sample.xsl', 'result.xml')) {
    print "SUCCESS, sample.xml was transformed by sample.xsl into result.xml";
    print ", result.xml has the following contents\n<br>\n";
    print "<pre>\n";
    readfile('result.xml');
    print "</pre>\n";
}
else {
    print "Sorry, sample.xml could not be transformed by sample.xsl into";
    print "  result.xml the reason is that " . xslt_error($xh) . " and the ";
    print "error code is " . xslt_errno($xh);
}

xslt_free($xh);

?>
```

Mentre questa funzionalità è importante, a volte, specialmente in un ambiente web, si vorrebbe avere la possibilità di scrivere a video il risultato direttamente. Quindi, se si omette il terzo argomento alla funzione **xslt_process()** (o si inserisce il valore NULL per l'argomento), lo script restituirà automaticamente il valore della trasformazione dell'XSLT, invece di scriverlo in un file...

Esempio 2. Uso di xslt_process() per trasformare un file XML e uno XSL in una variabile contenente i dati XML restituiti

```
<?php

// Allocate a new XSLT processor
$xh = xslt_create();

// Process the document, returning the result into the $result variable
$result = xslt_process($xh, 'sample.xml', 'sample.xsl');
if ($result) {
    print "SUCCESS, sample.xml was transformed by sample.xsl into the \$result";
    print " variable, the \$result variable has the following contents\n<br>\n";
    print "<pre>\n";
    print $result;
    print "</pre>\n";
}
else {
    print "Sorry, sample.xml could not be transformed by sample.xsl into";
    print "  the \$result variable the reason is that " . xslt_error($xh) .
    print " and the error code is " . xslt_errno($xh);
}

xslt_free($xh);

?>
```

I due casi sopra sono i due casi più semplici che ci sono quando c'è una trasformazione XSLT e c'è da dire che dovreste essere per la maggior parte delle volte in questi casi, ma, a volte, puoi prendere il tuo codice XML e XSLT da fonti esterne, come database e socket. In questi casi, avrai i dati XML e/o XSLT in una variabile -- nella produzione di applicazioni l'overhead per scaricare questo codice al file potrebbere essere eccessivo. Qui la sinstassi degli argomenti dell'XSLT ci aiuta. Invece dei file come argomenti XML e XSLT alla funzione **xslt_process()**, puoi specificare l' "argument place holders" il quale è poi sostituito dal valore dato nell'argomento dell'array (il quinto parametro della funzione **xslt_process()**). Di seguito c'è un esempio del processo di inserimento di codice XML e XSLT senza l'ausilio di file.

Esempio 3. Uso di xslt_process() per trasformare una variabile contenente dati XML e una variabile contenente dati XSL in una variabile contenente i dati XML risultanti

```
<?php
// $xml and $xsl contain the XML and XSL data

$arguments = array(
```

```
        '/_xml' => $xml,
        '/_xsl' => $xsl
);

// Allocate a new XSLT processor
$xh = xslt_create();

// Process the document
$result = xslt_process($xh, 'arg:/_xml', 'arg:/_xsl', NULL, $arguments);
if ($result) {
    print "SUCCESS, sample.xml was transformed by sample.xsl into the \$result";
    print " variable, the \$result variable has the following contents\n<br>\n";
    print "<pre>\n";
    print $result;
    print "</pre>\n";
}
else {
    print "Sorry, sample.xml could not be transformed by sample.xsl into";
    print "  the \$result variable the reason is that " . xslt_error($xh) .
    print " and the error code is " . xslt_errno($xh);
}
xslt_free($xh);
?>
```

Finalmente, l'ultimo argomento della funzione **xslt_process()** è dei parametri che vuoi passare al documento XSLT. Questi parametri possono avere poi accesso tramite i tuoi files XSL usando l'istruzione <xsl:param name="parameter_name">.

xslt_set_base (PHP 4 >= 4.0.5)

Imposta l'URI di base per tuttte le trasformazioni XSLT

void **xslt_set_base** (resource xh, string uri) \linebreak

Imposta l'URI di base per tutte le trasformazioni XSLT, l'URI di base è usato con le istruzioni Xpath per la risoluzione di document() e di altri comandi che accedono a risorse esterne.

xslt_set_encoding (PHP 4 >= 4.0.5)

Imposta l'encoding per il parsing dei documenti XML

void **xslt_set_encoding** (resource xh, string encoding) \linebreak

Imposta l'output encoding per le trasformazioni XSLT. Quando è usato il Sablotron backend quasta opzione è disponibile solo quando compili Sablotron con il supporto per l'encoding.

xslt_set_error_handler (PHP 4 >= 4.0.4)

Imposta un error handler per un processore XSLT

void **xslt_set_error_handler** (resource xh, mixed handler) \linebreak

Imposta una funzione di error handler per il processore XSLT dato da *xh*, questa funzione sarà richiamata quando ha luogo un errore nella trasformazione XSLT (questa funzione è anche richiamata per le nuove notizie dallo script).

xslt_set_log (PHP 4 >= 4.0.6)

Imposta il file di log per scrivere i messaggi di log

void **xslt_set_log** (resource xh, mixed log) \linebreak

xh

Un riferimento all'XSLT parser.

log

Questo parametro può essere un valore booleano che può essere settato su on o off oppure una stringa contenente il file di log incluso dei log di errore.

Questa funzione permette di impostare il file nel quale vuoi i messaggi di log di XSLT, i messaggi di log di XSLT sono differenti dagli altri messaggi di errore, nei quali messaggi di log non siano presenti i messaggi di errore ma piuttosto messaggi inerenti allo stato del processore di XSLT. Sono usati per il debugging dell'XSLT, quando qualcosa va storto.

Per default il logging è disabilitato, ma se vuoi abilitarlo devi prima richiamare **xslt_set_log()** con un parametro booleano il quale abilita il logging, poi se vuoi impostare il file di log per fare il debug, devi poi passargli una stringa contenente il nome del file:

Esempio 1. Uso delle features del Logging di XSLT

```
<?php

$xh = xslt_create();
xslt_set_log($xh, true);
xslt_set_log($xh, getcwd() . 'myfile.log');

$result = xslt_process($xh, 'dog.xml', 'pets.xsl');
print($result);

xslt_free($xh);
?>
```

xslt_set_sax_handler (4.0.3 - 4.0.6 only)

Imposta un handler SAX per un processore XSLT

void **xslt_set_sax_handler** (resource xh, array handler) \linebreak

Imposta una handler SAX sulla risorsa dell'handle data da *xh*. L'handler SAX può avere un array bi-dimensionale con il formato (tutti gli elementi al livello top sono opzionali):

```
array(
[document] =>
    array(
        start document handler,
        end document handler
    ),
[element] =>
    array(
        start element handler,
        end element handler
    ),
[namespace] =>
    array(
        start namespace handler,
        end namespace handler
    ),
[comment] => comment handler,
[pi] => processing instruction handler,
[character] => character data handler
)
```

xslt_set_sax_handlers (PHP 4 >= 4.0.6)

Imposta gli handler SAX da richiamare quando il document XML viene processato

void **xslt_set_sax_handlers** (resource processor, array handler) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

xslt_set_scheme_handler (4.0.5 - 4.0.6 only)

Imposta lo schema dell'handler per un processore XSLT

void **xslt_set_scheme_handler** (resource xh, array handler) \linebreak

Imposta lo schema dell'handler sulla risorsa dell'handle data da *xh*. Lo schema dell'handler può essere un array con il formato (tutti gli elementi sono opzionali):

```
array(
[get_all] => get all handler,
[open] => open handler,
[get] => get handler,
[put] => put handler,
[close] => close handler
)
```

xslt_set_scheme_handlers (PHP 4 >= 4.0.6)

Imposta lo schema degli handler per un processore XSLT

void **xslt_set_scheme_handlers** (resource processor, array handler) \linebreak

Attenzione

Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

CVI. YAZ functions

Introduction

This extension offers a PHP interface to the YAZ toolkit that implements the Z39.50 protocol for information retrieval. With this extension you can easily implement a Z39.50 origin (client) that searches or scans Z39.50 targets (servers) in parallel.

YAZ is available at http://www.indexdata.dk/yaz/. You can find news information, example scripts, etc. for this extension at http://www.indexdata.dk/phpyaz/.

The module hides most of the complexity of Z39.50 so it should be fairly easy to use. It supports persistent stateless connections very similar to those offered by the various SQL APIs that are available for PHP. This means that sessions are stateless but shared amongst users, thus saving the connect and initialize phase steps in most cases.

Installation

Compile YAZ and install it. Build PHP with your favourite modules and add option `--with-yaz`. Your task is roughly the following:

```
gunzip -c yaz-1.6.tar.gz|tar xf -
gunzip -c php-4.0.X.tar.gz|tar xf -
cd yaz-1.6
./configure --prefix=/usr
make
make install
cd ../php-4.0.X
./configure --with-yaz=/usr/bin
make
make install
```

Example

PHP/YAZ keeps track of connections with targets (Z-Associations). A positive integer represents the ID of a particular association.

Esempio 1. Parallel searching using YAZ()

The script below demonstrates the parallel searching feature of the API. When invoked with no arguments it prints a query form; else (arguments are supplied) it searches the targets as given in in array host.

```
$num_hosts = count ($host);
if (empty($term) || count($host) == 0) {
    echo '<form method="get">
    <input type="checkbox"
    name="host[]" value="bagel.indexdata.dk/gils">
        GILS test
    <input type="checkbox"
    name="host[]" value="localhost:9999/Default">
        local test
    <input type="checkbox" checked="1"
    name="host[]" value="z3950.bell-labs.com/books">
        BELL Labs Library
    <br>
    RPN Query:
    <input type="text" size="30" name="term">
    <input type="submit" name="action" value="Search">
    ';
} else {
    echo 'You searced for ' . htmlspecialchars($term) . '<br>';
    for ($i = 0; $i < $num_hosts; $i++) {
        $id[] = yaz_connect($host[$i]);
        yaz_syntax($id[$i],"sutrs");
        yaz_search($id[$i],"rpn",$term);
    }
    yaz_wait();
    for ($i = 0; $i < $num_hosts; $i++) {
        echo '<hr>' . $host[$i] . ":";
        $error = yaz_error($id[$i]);
        if (!empty($error)) {
            echo "Error: $error";
        } else {
            $hits = yaz_hits($id[$i]);
            echo "Result Count $hits";
        }
        echo '<dl>';
        for ($p = 1; $p <= 10; $p++) {
            $rec = yaz_record($id[$i],$p,"string");
            if (empty($rec)) continue;
            echo "<dt><b>$p</b></dt><dd>";
            echo ereg_replace("\n", "<br>\n",$rec);
            echo "</dd>";
        }
        echo '</dl>';
    }
}
```

yaz_addinfo (PHP 4)

Returns additional error information

int **yaz_addinfo** (int id) \linebreak

Returns additional error message for target (last request). An empty string is returned if last operation was a success or if no additional information was provided by the target.

yaz_ccl_conf (PHP 4 >= 4.0.5)

Configure CCL parser

int **yaz_ccl_conf** (int id, array config) \linebreak

This function configures the CCL query parser for a target with definitions of access points (CCL qualifiers) and their mapping to RPN. To map a specific CCL query to RPN afterwards call the yaz_ccl_parse() function. Each index of the array *config* is the name of a CCL field and the corresponding value holds a string that specifies a mapping to RPN. The mapping is a sequence of attribute-type, attribute-value pairs. Attribute-type and attribute-value is separated by an equal sign (=). Each pair is separated by white space.

Esempio 1. CCL configuration

In the example below, the CCL parser is configured to support three CCL fields: `ti`, `au` and `isbn`. Each field is mapped to their BIB-1 equivalent. It is assumed that variable `$id` is a target ID.

```
$field["ti"] = "1=4";
$field["au"] = "1=1";
$field["isbn"] = "1=7";
yaz_ccl_conf($id,$field);
```

yaz_ccl_parse (PHP 4 >= 4.0.5)

Invoke CCL Parser

int **yaz_ccl_parse** (int id, string query, array & result) \linebreak

This function invokes the CCL parser. It converts a given CCL FIND query to an RPN query which may be passed to the yaz_search() function to perform a search. To define a set of valid CCL fields call yaz_ccl_conf() prior to this function. If the supplied *query* was successfully converted to RPN, this function returns `TRUE`, and the index `rpn` of the supplied array *result* holds a valid RPN query. If the query could not be converted (because of invalid syntax, unknown field, etc.) this function returns `FALSE` and three indexes are set in the resulting array to indicate the cause of

failure: `errorcode`CCL error code (integer), `errorstring`CCL error string, and `errorpos`approximate position in query of failure (integer is character position).

yaz_close (PHP 4)

Closes a YAZ connection

int **yaz_close** (int id) \linebreak

Closes the Z-association given by *`id`*. The *`id`* is a target ID as returned by a previous yaz_connect() command.

yaz_connect (PHP 4)

Prepares for a connection and Z-association to a Z39.50 target.

int **yaz_connect** (string zurl [, mixed options]) \linebreak

This function returns a positive ID on success; zero on failure.

yaz_connect() prepares for a connection to a Z39.50 target. The zurl argument takes the form host[:port][/database]. If port is omitted 210 is used. If database is omitted Default is used. This function is non-blocking and doesn't attempt to establish a socket - it merely prepares a connect to be performed later when yaz_wait() is called.

If the second argument, *`options`*, is given as a string it is treated as the Z39.50 V2 authentication string (OpenAuth).

If *`options`* is given as an array the contents of the array serves as options. Note that array options are only supported for PHP 4.1.0 and later.

yaz_connect() options

user

Username for authentication.

group

Group for authentication.

password

Password for authentication.

cookie

Cookie for session (YAZ proxy).

proxy

Proxy for connection (YAZ proxy).

persistent

A boolean. If `TRUE` the connection is persistent; If `FALSE` the connection is not persistent. By default connections are persistent.

piggyback

A boolean. If `TRUE` piggyback is enabled for searches; If `FALSE` piggyback is disabled. By default piggyback is enabled. Enabling piggyback is more efficient and usually saves a network-round-trip for first time fetches of records. However, a few Z39.50 targets doesn't support piggyback or they ignore element set names. For those, piggyback should be disabled.

yaz_database (PHP 4 >= 4.0.6)

Specifies the databases within a session

int **yaz_database** (int id, string databases) \linebreak

This function specifies one or more databases to be used in search, retrieval, etc. - overriding databases specified in call to yaz_connect(). Multiple databases are separated by a plus sign `+`.

This function allows you to use different sets of databases within a session.

Returns `TRUE` on success; `FALSE` on error.

yaz_element (PHP 4)

Specifies Element-Set Name for retrieval

int **yaz_element** (int id, string elementset) \linebreak

This function is used in conjunction with yaz_search() and yaz_present() to specify the element set name for records to be retrieved. Most servers support `F` (full) and `B` (brief).

Returns `TRUE` on success; `FALSE` on error.

yaz_errno (PHP 4)

Returns error number

int **yaz_errno** (int id) \linebreak

Returns error for target (last request). A positive value is returned if the target returned a diagnostic code; a value of zero is returned if no errors occurred (success); negative value is returned for other errors (such as when the target closed connection, etc).

yaz_errno() should be called after network activity for each target - (after yaz_wait() returns) to determine the success or failure of the last operation (e.g. search).

yaz_error (PHP 4)

Returns error description

string **yaz_error** (int id) \linebreak

Returns error message for target (last request). An empty string is returned if last operation was a success.

yaz_error() returns an english text message corresponding to the last error number as returned by yaz_errno().

yaz_hits (PHP 4)

Returns number of hits for last search

int **yaz_hits** (int id) \linebreak

yaz_hits() returns number of hits for last search.

yaz_itemorder (PHP 4 >= 4.0.5)

Prepares for Z39.50 Item Order with an ILL-Request package

int **yaz_itemorder** (array args) \linebreak

This function prepares for an Extended Services request using the Profile for the Use of Z39.50 Item Order Extended Service to Transport ILL (Profile/1). See this (http://www.nlc-bnc.ca/iso/ill/stanprf.htm) and the specification (http://www.nlc-bnc.ca/iso/ill/document/standard/z-ill-1a.pdf). The args parameter must be a hash array with information about the Item Order request to be sent. The key of the hash is the name of the corresponding ASN.1 tag path. For example, the ISBN below the Item-ID has the key item-id,ISBN.

The ILL-Request parameters are:

protocol-version-num
transaction-id,initial-requester-id,person-or-institution-symbol,person
transaction-id,initial-requester-id,person-or-institution-symbol,institution
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-person
transaction-id,initial-requester-id,name-of-person-or-institution,name-of-institution
transaction-id,transaction-group-qualifier
transaction-id,transaction-qualifier
transaction-id,sub-transaction-qualifier
service-date-time,this,date
service-date-time,this,time
service-date-time,original,date
service-date-time,original,time
requester-id,person-or-institution-symbol,person
requester-id,person-or-institution-symbol,institution
requester-id,name-of-person-or-institution,name-of-person
requester-id,name-of-person-or-institution,name-of-institution

responder-id,person-or-institution-symbol,person
responder-id,person-or-institution-symbol,institution
responder-id,name-of-person-or-institution,name-of-person
responder-id,name-of-person-or-institution,name-of-institution
transaction-type
delivery-address,postal-address,name-of-person-or-institution,name-of-person
delivery-address,postal-address,name-of-person-or-institution,name-of-institution
delivery-address,postal-address,extended-postal-delivery-address
delivery-address,postal-address,street-and-number
delivery-address,postal-address,post-office-box
delivery-address,postal-address,city
delivery-address,postal-address,region
delivery-address,postal-address,country
delivery-address,postal-address,postal-code
delivery-address,electronic-address,telecom-service-identifier
delivery-address,electronic-address,telecom-service-addreess
billing-address,postal-address,name-of-person-or-institution,name-of-person
billing-address,postal-address,name-of-person-or-institution,name-of-institution
billing-address,postal-address,extended-postal-delivery-address
billing-address,postal-address,street-and-number
billing-address,postal-address,post-office-box
billing-address,postal-address,city
billing-address,postal-address,region
billing-address,postal-address,country
billing-address,postal-address,postal-code
billing-address,electronic-address,telecom-service-identifier
billing-address,electronic-address,telecom-service-addreess
ill-service-type
requester-optional-messages,can-send-RECEIVED
requester-optional-messages,can-send-RETURNED
requester-optional-messages,requester-SHIPPED
requester-optional-messages,requester-CHECKED-IN
search-type,level-of-service
search-type,need-before-date
search-type,expiry-date
search-type,expiry-flag
place-on-hold
client-id,client-name
client-id,client-status
client-id,client-identifier
item-id,item-type
item-id,call-number
item-id,author
item-id,title
item-id,sub-title
item-id,sponsoring-body
item-id,place-of-publication
item-id,publisher
item-id,series-title-number
item-id,volume-issue
item-id,edition
item-id,publication-date

item-id,publication-date-of-component
item-id,author-of-article
item-id,title-of-article
item-id,pagination
item-id,ISBN
item-id,ISSN
item-id,additional-no-letters
item-id,verification-reference-source
copyright-complicance
retry-flag
forward-flag
requester-note
forward-note

There are also a few parameters that are part of the Extended Services Request package and the ItemOrder package:

package-name
user-id
contact-name
contact-phone
contact-email
itemorder-item

yaz_present (PHP 4 >= 4.0.5)

Prepares for retrieval (Z39.50 present).

int **yaz_present** (void) \linebreak

This function prepares for retrieval of records after a successful search. The yaz_range() should be called prior to this function to specify the range of records to be retrieved.

yaz_range (PHP 4)

Specifies the maximum number of records to retrieve

int **yaz_range** (int id, int start, int number) \linebreak

This function is used in conjunction with yaz_search() to specify the maximum number of records to retrieve (number) and the first record position (start). If this function is not invoked (only yaz_search()) start is set to 1 and number is set to 10.

Returns TRUE on success; FALSE on error.

yaz_record (PHP 4)

Returns a record

int **yaz_record** (int id, int pos, string type) \linebreak

Returns record at position or empty string if no record exists at given position.

The **yaz_record()** function inspects a record in the current result set at the position specified. If no database record exists at the given position an empty string is returned. The argument, type, specifies the form of the returned record. If type is "string" the record is returned in a string representation suitable for printing (for XML and SUTRS). If type is "array" the record is returned as an array representation (for structured records).

yaz_scan (PHP 4 >= 4.0.5)

Prepares for a scan

int **yaz_scan** (int id, string type, string startterm [, array flags]) \linebreak

This function prepares for a Z39.50 Scan Request. Argument *id* specifies target ID. Starting term point for the scan is given by *startterm*. The form in which is the starting term is specified is given by *type*. Currently type `rpn` is supported. The optional *flags* specifies additional information to control the behaviour of the scan request. Three indexes are currently read from the flags: `number` (number of terms requested), `position` (preferred position of term) and `stepSize` (preferred step size). To actually tranfer the Scan Request to the target and receive the Scan Response, yaz_wait() must be called. Upon completion of yaz_wait() call yaz_error() and yaz_scan_result() to handle the response.

The syntax of *startterm* is similar to the RPN query as described in yaz_search(). The startterm consists of zero or more `@attr`-operator specifications, then followed by exactly one token.

Esempio 1. PHP function that scans titles

```
function scan_titles($id, $starterm) {
  yaz_scan($id,"rpn", "@attr 1=4 " . $starterm);
  yaz_wait();
  $errno = yaz_errno($id);
  if ($errno == 0) {
    $ar = yaz_scan_result($id,&$options);
    echo ’Scan ok; ’;
    $ar = yaz_scan_result($id, &$options);
    while(list($key,$val)=each($options)) {
       echo "$key = $val  ";
    }
    echo ’<br><table><tr><td>’;
    while(list($key,list($k, $term, $tcount))=each($ar)) {
      if (empty($k)) continue;
      echo "<tr><td>$term</td><td>";
      echo $tcount;
      echo "</td></tr>";
    }
```

```
    echo '</table>';
  } else {
    echo "Scan failed. Error: " . yaz_error($id) . "<br>";
  }
}
```

yaz_scan_result (PHP 4 >= 4.0.5)

Returns Scan Response result

array **yaz_scan_result** (int id [, array & result]) \linebreak

Given a target ID this function returns and array with terms as received from the target in the last Scan Response. This function returns an array (0..n-1) where n is the number of terms returned. Each value is a pair where first item is term, second item is result-count. If the *`result`* is given it will be modified to hold additional information taken from the Scan Response: `number` (number of entries returned), `stepsize` (Step-size), `position` (position of term), `status` (Scan Status).

yaz_search (PHP 4)

Prepares for a search

int **yaz_search** (int id, string type, string query) \linebreak

yaz_search() prepares for a search on the target with given id. The type represents the query type - only "rpn" is supported now in which case the third argument specifies a Type-1 query (RPN). Like yaz_connect() this function is non-blocking and only prepares for a search to be executed later when yaz_wait() is called.

The RPN query is a textual represenation of the Type-1 query as defined by the Z39.50 standard. However, in the text representation as used by YAZ a prefix notation is used, that is the operater precedes the operands. The query string is a sequence of tokens where white space is ignored is ignored unless surrounded by double quotes. Tokens beginning with an at-character (`@`) are considered operators, otherwise they are treated as search terms.

Tabella 1. RPN Operators

Syntax	Description
`@and query1 query2`	intersection of query1 and query2
`@or query1 query2`	union of query1 and query2
`@not query1 query2`	query1 and not query2
`@set name`	result set reference

Syntax	Description
`@attrset set query`	specifies attribute-set for query. This construction is only allowed once - in the beginning of the whole query
`@attr set type=value query`	applies attribute to query. The type and value are integers specifying the attribute-type and attribute-value respectively. The set, if given, specifies the attribute-set.

Esempio 1. Query Examples

Query

`computer`
matches documents where "computer" occur. No attributes are specified.

The Query

`"donald knuth"`
matches documents where "donald knuth" occur.

For the query

`@attr 1=4 art`
attribute type is 1 (Bib-1 use), attribute value is 4 Title), so this should match documents where `art` occur in the title.

Another more complex one:

`@attrset gils @and @attr 1=4 art @attr 1=1003 "donald knuth"`
The query as a whole uses the GILS attributeset. The query matches documents where `art` occur in the title and in which `donald knuth` occur in the author.

yaz_sort (PHP 4 >= 4.1.0)

Sets sorting criteria

int **yaz_sort** (int id, string criteria) \linebreak

This function sets sorting criteria and enables Z39.50 Sort. Use this function together with yaz_search() or yaz_present(). Using this function alone doesn't have any effect. If used in conjunction with yaz_search() a Z39.50 Sort will be sent after a search response has been received and before any records are retrieved with Z39.50 Present. The *`criteria`* takes the form

`field1 flags1 field2 flags2` ...

where field1 specifies primary attributes for sort, field2 seconds, etc.. The field specifies either numerical attribute combinations consisting of type=value pairs separated by comma (e.g. `1=4,2=1`) ; or the field may specify a plain string criteria (e.g. `title`. The flags is a sequnce of the following characters which may not be separated by any white space.

Sort Flags

`a`

Sort ascending

`d`

Sort descending

`i`

Case insensitive sorting

`s`

Case sensitive sorting

Esempio 1. Sort Criterias

To sort on Bib1 attribute title, case insensitive, and ascending you'd use the following sort criteria:

```
1=4 ia
```

If the secondary sorting criteria should be author, case sensitive and ascending you'd use:

```
1=4 ia 1=1003 sa
```

yaz_syntax (PHP 4)

Specifies the preferred record syntax for retrieval.

int **yaz_syntax** (int id, string syntax) \linebreak

The syntax is specified as an OID (Object Identifier) in a raw dot-notation (like `1.2.840.10003.5.10`) or as one of the known registered record syntaxes (sutrs, usmarc, grs1, xml, etc.). This function is used in conjunction with yaz_search() and yaz_present() to specify the preferred record syntax for retrieval.

yaz_wait (PHP 4)

Wait for Z39.50 requests to complete

int **yaz_wait** ([array options]) \linebreak

This function carries out networked (blocked) activity for outstanding requests which have been prepared by the functions yaz_connect(), yaz_search(), yaz_present(), yaz_scan() and yaz_itemorder(). **yaz_wait()** returns when all targets have either completed all requests or aborted (in case of errors).

If the *`options`* array is given that holds options that change the behaviour of **yaz_wait()**.

`timeout`

Sets timeout in seconds. If a target hasn't responded within the timeout it is considered dead and **yaz_wait()** returns. The default value for timeout is 15 seconds.

CVII. YP/NIS Functions

NIS (formerly called Yellow Pages) allows network management of important administrative files (e.g. the password file). For more information refer to the NIS manpage and Introduction to YP/NIS (http://www.desy.de/~sieversm/ypdoku/ypdoku/ypdoku.html). There is also a book called Managing NFS and NIS (http://www.oreilly.com/catalog/nfs/noframes.html) by Hal Stern.

To get these functions to work, you have to configure PHP with `--with-yp`(PHP 3) or `--enable-yp`(PHP 4).

yp_all (PHP 4 >= 4.0.6)

Traverse the map and call a function on each entry

void **yp_all** (string domain, string map, string callback) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

yp_cat (PHP 4 >= 4.0.6)

Return an array containing the entire map

array **yp_cat** (string domain, string map) \linebreak

> **Attenzione**
>
> Questa funzione, al momento non è documentata; è disponibile soltanto la lista degli argomenti.

yp_err_string (PHP 4 >= 4.0.6)

Returns the error string associated with the previous operation

string **yp_err_string** (void) \linebreak

yp_err_string() returns the error message associated with the previous operation. Useful to indicate what exactly went wrong.

Esempio 1. Example for NIS errors

```
<?php
    echo "Error: " . yp_err_string();
?>
```

See also yp_errno().

yp_errno (PHP 4 >= 4.0.6)

Returns the error code of the previous operation

int **yp_errno** (void) \linebreak

yp_errno() returns the error code of the previous operation.

Possible errors are:

1 args to function are bad
2 RPC failure - domain has been unbound
3 can’t bind to server on this domain
4 no such map in server’s domain
5 no such key in map
6 internal yp server or client error
7 resource allocation failure
8 no more records in map database
9 can’t communicate with portmapper
10 can’t communicate with ypbind
11 can’t communicate with ypserv
12 local domain name not set
13 yp database is bad
14 yp version mismatch
15 access violation
16 database busy

See also yp_err_string().

yp_first (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Returns the first key-value pair from the named map

array **yp_first** (string domain, string map) \linebreak

yp_first() returns the first key-value pair from the named map in the named domain, otherwise FALSE.

Esempio 1. Example for the NIS first

```
<?php
$entry = yp_first($domain, "passwd.byname");
$key = $entry ["key"];
$value = $entry ["value"];
echo "First entry in this map has key " . $key . " and value " . $value;
?>
```

See also yp-get-default-domain()

yp_get_default_domain (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Fetches the machine's default NIS domain

int **yp_get_default_domain** (void) \linebreak

yp_get_default_domain() returns the default domain of the node or `FALSE`. Can be used as the domain parameter for successive NIS calls.

A NIS domain can be described a group of NIS maps. Every host that needs to look up information binds itself to a certain domain. Refer to the documents mentioned at the beginning for more detailed information.

Esempio 1. Example for the default domain

```
<?php
$domain = yp_get_default_domain();
echo "Default NIS domain is: " . $domain;
?>
```

yp_master (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Returns the machine name of the master NIS server for a map

string **yp_master** (string domain, string map) \linebreak

yp_master() returns the machine name of the master NIS server for a map.

Esempio 1. Example for the NIS master

```
<?php
$number = yp_master ($domain, $mapname);
echo "Master for this map is: " . $master;
?>
```

See also yp-get-default-domain().

yp_match (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Returns the matched line

string **yp_match** (string domain, string map, string key) \linebreak

yp_match() returns the value associated with the passed key out of the specified map or FALSE. This key must be exact.

Esempio 1. Example for NIS match

```
<?php
$entry = yp_match ($domain, "passwd.byname", "joe");
echo "Matched entry is: " . $entry;
?>
```

In this case this could be: joe:##joe:11111:100:Joe User:/home/j/joe:/usr/local/bin/bash

See also yp-get-default-domain()

yp_next (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Returns the next key-value pair in the named map.

array **yp_next** (string domain, string map, string key) \linebreak

yp_next() returns the next key-value pair in the named map after the specified key or FALSE.

Esempio 1. Example for NIS next

```
<?php
$entry = yp_next ($domain, "passwd.byname", "joe");

if (!$entry) {
echo "No more entries found\n";
    <!-- echo yp_errno() . ": " . yp_err_string(); -->
}

$key = key ($entry);

echo "The next entry after joe has key " . $key
      . " and value " . $entry[$key];
?>
```

See also yp-get-default-domain().

yp_order (PHP 3>= 3.0.7, PHP 4 >= 4.0.0)

Returns the order number for a map

int **yp_order** (string domain, string map) \linebreak

yp_order() returns the order number for a map or FALSE.

Esempio 1. Example for the NIS order

```
<?php
    $number = yp_order($domain,$mapname);
    echo "Order number for this map is: " . $number;
?>
```

See also yp-get-default-domain().

CVIII. Funzioni per File Zip (Accesso di Sola Lettura)

Questo modulo usa le funzioni ZZIPlib (http://zziplib.sourceforge.net/) della libreria di Guido Draheim per leggere archivi ZIP compressi e i file in essi contenuti.

Notare che ZZIPlib rende disponbili solo un sottogruppo di quelle funzioni disponibili in una implementazione completa dell'algoritmo di compressione ZIP e può solamente leggere i file in formato ZIP. Una normale utility ZIP è richiesta per creare i file ZIP letti da questa libreria.

Il supporto Zip all'interno di PHP non è abilitato di default. Sarà necessario usare l'opzione di configurazione --with-zip durante la compilazione di PHP per abilitare tale supporto. Questo modulo richiede ZZIPlib versione >= 0.10.6.

> **Nota:** Il supporto Zip precedentemente alla versione 4.1.0 di PHP è sperimentale. Questa sezione riflette l'estensione Zip così come essa esiste in PHP 4.1.0 e successivi.

Esempio di Utilizzo

Questo esempio apre un archivio ZIP, legge tutti i file presenti nell'archivio e stampa il contenuto. L'archivio `test2.zip` usato in questo esempio è uno degli archivi dimostrativi presenti nella distribuzione di ZZIPlib.

Esempio 1. Esempio di Utilizzo Zip

```
<?php

$zip = zip_open("/tmp/test2.zip");

if ($zip) {

    while ($zip_entry = zip_read($zip)) {
        echo "Nome:                     " . zip_entry_name($zip_entry) . "\n";
        echo "Dimensione File:          " . zip_entry_filesize($zip_entry) . "\n";
        echo "Dimensione Compressa:     " . zip_entry_compressedsize($zip_entry) . "\n";
        echo "Metodo di Compressione:   " . zip_entry_compressionmethod($zip_entry) . "\n"

        if (zip_entry_open($zip, $zip_entry, "r")) {
            echo "Contenuto File:\n";
            $buf = zip_entry_read($zip_entry, zip_entry_filesize($zip_entry));
            echo "$buf\n";

            zip_entry_close($zip_entry);
        }
        echo "\n";

    }

    zip_close($zip);

}
```

```
?>
```

zip_close (PHP 4 >= 4.1.0)

Chiude un archivio Zip

void **zip_close** (resource zip) \linebreak

Chiude un file di archivio Zip. Il parametro `zip` deve essere un archivio zip precedentemente aperto da zip_open().

Questa funzione non restituisce alcun valore.

Vedere anche zip_open() e zip_read().

zip_entry_close (PHP 4 >= 4.1.0)

Chiude il puntatore a una directory

void **zip_entry_close** (resource zip_entry) \linebreak

Chiude il puntatore specificato da `zip_entry`. Il parametro `zip_entry` deve essere un puntatore valido aperto in precedenza da zip_entry_open().

Questa funzione non restituisce alcun valore.

Vedere anche zip_entry_open() e zip_entry_read().

zip_entry_compressedsize (PHP 4 >= 4.1.0)

Ottiene la dimensione compressa di una Directory

int **zip_entry_compressedsize** (resource zip_entry) \linebreak

Restituisce la dimensione compressa della voce directory specificata da `zip_entry`. Il parametro `zip_entry` è un riferimento alla voce directory valido restituito da zip_read().

Vedere anche zip_open() e zip_read().

zip_entry_compressionmethod (PHP 4 >= 4.1.0)

Ottiene il metodo di compressione di una voce directory

string **zip_entry_compressionmethod** (resource zip_entry) \linebreak

Restituisce il metodo di compressione di una voce directory specificata da `zip_entry`. Il parametro `zip_entry` è una voce directory valida restituita da zip_read().

Vedere anche zip_open() e zip_read().

zip_entry_filesize (PHP 4 >= 4.1.0)

Ottiene la dimensione attuale di una directory

int **zip_entry_filesize** (resource zip_entry) \linebreak

Restituisce la dimensione attuale della directory specificata da *zip_entry*. Il parametro *zip_entry* è una voce directory valida restituita da zip_read().

Vedere anche zip_open() e zip_read().

zip_entry_name (PHP 4 >= 4.1.0)

Ottiene il nome di una directory

string **zip_entry_name** (resource zip_entry) \linebreak

Restituisce il nome di una voce directory specificata da *zip_entry*. Il parametro *zip_entry* è una voce directory valida restituita da zip_read().

Vedere anche zip_open() e zip_read().

zip_entry_open (PHP 4 >= 4.1.0)

Apre una voce directory in lettura

bool **zip_entry_open** (resource zip, resource zip_entry [, string mode]) \linebreak

Apre una voce directory in lettura in un file zip. Il parametro *zip* è un resource handle valido restituito da zip_open(). Il parametro *zip_entry* è una voce directory restituita da zip_read(). Il parametro opzionale *mode* può essere uno qualunque dei modi specificati nella documentazione relativa a fopen().

> **Nota:** Attualmente, *mode* viene ignorato e viene sempre usato `"rb"`. Questo è dato dal fatto che il supporto zip in PHP è di tipo sola lettura. Fare riferimento a fopen() per una spiegazione dei vari modi, incluso `"rb"`.

Restituisce `TRUE` in caso di successo, `FALSE` in caso di fallimento.

> **Nota:** A differenza di fopen() e altre funzioni simili, il valore restituito da **zip_entry_open()** indica solo il risultato dell'operazione e non è necessario per leggere o chiudere la voce relativa alla directory.

Vedere anche zip_entry_read() e zip_entry_close().

zip_entry_read (PHP 4 >= 4.1.0)

Legge da una directory aperta

string **zip_entry_read** (resource zip_entry [, int length]) \linebreak

Legge fino a *length* byte da una directory aperta. Se *length* non è specificato, allora **zip_entry_read()** tenterà di leggere 1024 byte. Il parametro *zip_entry* è una voce directory valida restituita da zip_read().

> **Nota:** Il parametro *length* deve essere la lunghezza non compressa che si desidera leggere.

Restituisce i dati letti, o FALSE se viene raggiunta la fine del file.

Vedere anche zip_entry_open(), zip_entry_close() e zip_entry_filesize().

zip_open (PHP 4 >= 4.1.0)

Apre un archivio zip

resource **zip_open** (string filename) \linebreak

Apre un nuovo archivio zip in lettura. Il parametro *filename* è il nome del file dell'archivio zip che si desidera aprire.

Restituisce un resource handle da usarsi di seguito con zip_read() e zip_close() o restituisce FALSE se *filename* non esiste.

Vedere anche zip_read() e zip_close().

zip_read (PHP 4 >= 4.1.0)

Legge la prossima voce in un archivio file zip

resource **zip_read** (resource zip) \linebreak

Legge il prossimo file in un archivio su file di tipo zip. Il parametro *zip* deve essere un archivio zip precedentemente aperto da zip_open().

Restituisce un puntatore alla risorsa directory da usarsi successivamente con le varie funzioni **zip_entry_...()**.

Vedere anche zip_open(), zip_close(), zip_entry_open() e zip_entry_read().

CIX. Zlib Compression Functions

This module uses the functions of zlib (http://www.gzip.org/zlib/) by Jean-loup Gailly and Mark Adler to transparently read and write gzip (.gz) compressed files. You have to use a zlib version >= 1.0.9 with this module.

This module contains versions of most of the filesystem functions which work with gzip-compressed files (and uncompressed files, too, but not with sockets).

> **Nota:** Version 4.0.4 introduces a fopen-wrapper for .gz-files, so that you can use a special 'zlib:' URL to access compressed files transparently using the normal f*() file access functions if you prepend the filename or path with a 'zlib:' prefix when calling fopen().
>
> In version 4.3.0, this special prefix has been changed to 'zlib://' to prevent ambiguities with filenames containing ':'.
>
> This feature requires a C runtime library that provides the `fopencookie()` function. To my current knowledge the GNU libc is the only library that provides this feature.

Small code example

Opens a temporary file and writes a test string to it, then it prints out the content of this file twice.

Esempio 1. Small Zlib Example

```
<?php

$filename = tempnam ('/tmp', 'zlibtest').'.gz';
print "<html>\n<head></head>\n<body>\n<pre>\n";
$s = "Only a test, test, test, test, test, test, test, test!\n";

// open file for writing with maximum compression
$zp = gzopen($filename, "w9");

// write string to file
gzwrite($zp, $s);

// close file
gzclose($zp);

// open file for reading
$zp = gzopen($filename, "r");

// read 3 char
print gzread($zp, 3);

// output until end of the file and close it.
gzpassthru($zp);

print "\n";

// open file and print content (the 2nd time).
if (readgzfile($filename) != strlen($s)) {
```

```
        echo "Error with zlib functions!";
}
unlink($filename);
print "</pre>\n</h1></body>\n</html>\n";

?>
```

gzclose (PHP 3, PHP 4 >= 4.0.0)

Close an open gz-file pointer

int **gzclose** (int zp) \linebreak

The gz-file pointed to by zp is closed.

Returns TRUE on success and FALSE on failure.

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

gzcompress (PHP 4)

Compress a string

string **gzcompress** (string data [, int level]) \linebreak

This function returns a compressed version of the input *data* using the ZLIB data format, or FALSE if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression.

For details on the ZLIB compression algorithm see the document "ZLIB Compressed Data Format Specification version 3.3 (http://www.ietf.org/rfc/rfc1950.txt)" (RFC 1950).

> **Nota:** This is *not* the same as gzip compression, which includes some header data. See gzencode() for gzip compression.

See also gzdeflate(), gzinflate(), gzuncompress(), gzencode().

gzdeflate (PHP 4 >= 4.0.4)

Deflate a string

string **gzdeflate** (string data [, int level]) \linebreak

This function returns a compressed version of the input *data* using the DEFLATE data format, or FALSE if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression.

For details on the DEFLATE compression algorithm see the document "DEFLATE Compressed Data Format Specification version 1.3 (http://www.ietf.org/rfc/rfc1951.txt)" (RFC 1951).

See also gzinflate(), gzcompress(), gzuncompress(), gzencode().

gzencode (PHP 4 >= 4.0.4)

Create a gzip compressed string

string **gzencode** (string data [, int level [, int encoding_mode]]) \linebreak

This function returns a compressed version of the input *data* compatible with the output of the **gzip** program, or FALSE if an error is encountered. The optional parameter *level* can be given as 0 for no compression up to 9 for maximum compression, if not given the default compression level will be the default compression level of the zlib library.

You can also give FORCE_GZIP (the default) or FORCE_DEFLATE as optional third paramter *encoding_mode*. If you use FORCE_DEFLATE, you get a standard zlib deflated string (inclusive zlib headers) after the gzip file header but without the trailing crc32 checksum.

> **Nota:** *level* was added in PHP 4.2, before PHP 4.2 **gzencode()** only had the *data* and (optional) *encoding_mode* parameters..

The resulting data contains the appropriate headers and data structure to make a standard .gz file, e.g.:

Esempio 1. Creating a gzip file

```
<?php
    $data = implode("", file("bigfile.txt"));
    $gzdata = gzencode($data, 9);
    $fp = fopen("bigfile.txt.gz", "w");
    fwrite($fp, $gzdata);
    fclose($fp);
?>
```

For more information on the GZIP file format, see the document: GZIP file format specification version 4.3 (http://www.ietf.org/rfc/rfc1952.txt) (RFC 1952).

See also gzcompress(). gzuncompress(), gzdeflate(), gzinflate().

gzeof (PHP 3, PHP 4 >= 4.0.0)

Test for end-of-file on a gz-file pointer

int **gzeof** (int zp) \linebreak

Returns TRUE if the gz-file pointer is at EOF or an error occurs; otherwise returns FALSE.

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

gzfile (PHP 3, PHP 4 >= 4.0.0)

Read entire gz-file into an array

array **gzfile** (string filename [, int use_include_path]) \linebreak

Identical to readgzfile(), except that **gzfile()** returns the file in an array.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also readgzfile(), and gzopen().

gzgetc (PHP 3, PHP 4 >= 4.0.0)

Get character from gz-file pointer

string **gzgetc** (int zp) \linebreak

Returns a string containing a single (uncompressed) character read from the file pointed to by zp. Returns FALSE on EOF (as does gzeof()).

The gz-file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), and gzgets().

gzgets (PHP 3, PHP 4 >= 4.0.0)

Get line from file pointer

string **gzgets** (int zp, int length) \linebreak

Returns a (uncompressed) string of up to length - 1 bytes read from the file pointed to by fp. Reading ends when length - 1 bytes have been read, on a newline, or on EOF (whichever comes first).

If an error occurs, returns FALSE.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), gzgetc(), and fgets().

gzgetss (PHP 3, PHP 4 >= 4.0.0)

Get line from gz-file pointer and strip HTML tags

string **gzgetss** (int zp, int length [, string allowable_tags]) \linebreak

Identical to gzgets(), except that **gzgetss()** attempts to strip any HTML and PHP tags from the text it reads.

You can use the optional third parameter to specify tags which should not be stripped.

Nota: *`Allowable_tags`* was added in PHP 3.0.13, PHP4B3.

See also gzgets(), gzopen(), and strip_tags().

gzinflate (PHP 4 >= 4.0.4)

Inflate a deflated string

string **gzinflate** (string data [, int length]) \linebreak

This function takes *data* compressed by gzdeflate() and returns the original uncompressed data or FALSE on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *data* or more than the optional parameter *length*.

See also gzcompress(). gzuncompress(), gzdeflate(), gzencode().

gzopen (PHP 3, PHP 4 >= 4.0.0)

Open gz-file

int **gzopen** (string filename, string mode [, int use_include_path]) \linebreak

Opens a gzip (.gz) file for reading or writing. The mode parameter is as in fopen() ("rb" or "wb") but can also include a compression level ("wb9") or a strategy: 'f' for filtered data as in "wb6f", 'h' for Huffman only compression as in "wb1h". (See the description of deflateInit2 in zlib.h for more information about the strategy parameter.)

gzopen() can be used to read a file which is not in gzip format; in this case gzread() will directly read from the file without decompression.

gzopen() returns a file pointer to the file opened, after that, everything you read from this file descriptor will be transparently decompressed and what you write gets compressed.

If the open fails, the function returns FALSE.

You can use the optional third parameter and set it to "1", if you want to search for the file in the include_path, too.

Esempio 1. gzopen() Example

```
$fp = gzopen ("/tmp/file.gz", "r");
```

See also gzclose().

gzpassthru (PHP 3, PHP 4 >= 4.0.0)

Output all remaining data on a gz-file pointer

int **gzpassthru** (int zp) \linebreak

Reads to EOF on the given gz-file pointer and writes the (uncompressed) results to standard output.

If an error occurs, returns `FALSE`.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

The gz-file is closed when **gzpassthru()** is done reading it (leaving *zp* useless).

gzputs (PHP 3, PHP 4 >= 4.0.0)

Write to a gz-file pointer

int **gzputs** (int zp, string str [, int length]) \linebreak

gzputs() is an alias to gzwrite(), and is identical in every way.

gzread (PHP 3, PHP 4 >= 4.0.0)

Binary-safe gz-file read

string **gzread** (int zp, int length) \linebreak

gzread() reads up to *length* bytes from the gz-file pointer referenced by *zp*. Reading stops when *length* (uncompressed) bytes have been read or EOF is reached, whichever comes first.

```
// get contents of a gz-file into a string
$filename = "/usr/local/something.txt.gz";
$zd = gzopen ($filename, "r");
$contents = gzread ($zd, 10000);
gzclose ($zd);
```

See also gzwrite(), gzopen(), gzgets(), gzgetss(), gzfile(), and gzpassthru().

gzrewind (PHP 3, PHP 4 >= 4.0.0)

Rewind the position of a gz-file pointer

int **gzrewind** (int zp) \linebreak

Sets the file position indicator for zp to the beginning of the file stream.

If an error occurs, returns 0.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzseek() and gztell().

gzseek (PHP 3, PHP 4 >= 4.0.0)

Seek on a gz-file pointer

int **gzseek** (int zp, int offset) \linebreak

Sets the file position indicator for the file referenced by zp to offset bytes into the file stream. Equivalent to calling (in C) `gzseek( zp, offset, SEEK_SET )`.

If the file is opened for reading, this function is emulated but can be extremely slow. If the file is opened for writing, only forward seeks are supported; gzseek then compresses a sequence of zeroes up to the new starting position.

Upon success, returns 0; otherwise, returns -1. Note that seeking past EOF is not considered an error.

See also gztell() and gzrewind().

gztell (PHP 3, PHP 4 >= 4.0.0)

Tell gz-file pointer read/write position

int **gztell** (int zp) \linebreak

Returns the position of the file pointer referenced by *`zp`*; i.e., its offset into the file stream.

If an error occurs, returns `FALSE`.

The file pointer must be valid, and must point to a file successfully opened by gzopen().

See also gzopen(), gzseek() and gzrewind().

gzuncompress (PHP 4)

Uncompress a deflated string

string **gzuncompress** (string data [, int length]) \linebreak

This function takes *`data`* compressed by gzcompress() and returns the original uncompressed data or `FALSE` on error. The function will return an error if the uncompressed data is more than 256 times the length of the compressed input *`data`* or more than the optional parameter *`length`*.

See also gzdeflate(), gzinflate(), gzcompress(), gzencode().

gzwrite (PHP 3, PHP 4 >= 4.0.0)

Binary-safe gz-file write

int **gzwrite** (int zp, string string [, int length]) \linebreak

gzwrite() writes the contents of *`string`* to the gz-file stream pointed to by *`zp`*. If the *`length`* argument is given, writing will stop after *`length`* (uncompressed) bytes have been written or the end of *`string`* is reached, whichever comes first.

Note that if the *`length`* argument is given, then the magic_quotes_runtime configuration option will be ignored and no slashes will be stripped from *`string`*.

See also gzread(), gzopen(), and gzputs().

readgzfile (PHP 3, PHP 4 >= 4.0.0)

Output a gz-file

int **readgzfile** (string filename [, int use_include_path]) \linebreak

Reads a file, decompresses it and writes it to standard output.

readgzfile() can be used to read a file which is not in gzip format; in this case **readgzfile()** will directly read from the file without decompression.

Returns the number of (uncompressed) bytes read from the file. If an error occurs, `FALSE` is returned and unless the function was called as `@readgzfile`, an error message is printed.

The file *`filename`* will be opened from the filesystem and its contents written to standard output.

You can use the optional second parameter and set it to "1", if you want to search for the file in the include_path, too.

See also gzpassthru(), gzfile(), and gzopen().

Parte V. Extending PHP 4.0

Capitolo 25. Overview

"Extending PHP" is easier said than done. PHP has evolved to a full-fledged tool consisting of a few megabytes of source code, and to hack a system like this quite a few things have to be learned and considered. When structuring this chapter, we finally decided on the "learn by doing" approach. This is not the most scientific and professional approach, but the method that's the most fun and gives the best end results. In the following sections, you'll learn quickly how to get the most basic extensions to work almost instantly. After that, you'll learn about Zend's advanced API functionality. The alternative would have been to try to impart the functionality, design, tips, tricks, etc. as a whole, all at once, thus giving a complete look at the big picture before doing anything practical. Although this is the "better" method, as no dirty hacks have to be made, it can be very frustrating as well as energy- and time-consuming, which is why we've decided on the direct approach.

Note that even though this chapter tries to impart as much knowledge as possible about the inner workings of PHP, it's impossible to really give a complete guide to extending PHP that works 100% of the time in all cases. PHP is such a huge and complex package that its inner workings can only be understood if you make yourself familiar with it by practicing, so we encourage you to work with the source.

What Is Zend? and What Is PHP?

The name *Zend* refers to the language engine, PHP's core. The term *PHP* refers to the complete system as it appears from the outside. This might sound a bit confusing at first, but it's not that complicated (see Figura 25-1). To implement a Web script interpreter, you need three parts:

1. The *interpreter* part analyzes the input code, translates it, and executes it.
2. The *functionality* part implements the functionality of the language (its functions, etc.).
3. The *interface* part talks to the Web server, etc.

Zend takes part 1 completely and a bit of part 2; PHP takes parts 2 and 3. Together they form the complete PHP package. Zend itself really forms only the language core, implementing PHP at its very basics with some predefined functions. PHP contains all the modules that actually create the language's outstanding capabilities.

Figura 25-1. The internal structure of PHP.

The following sections discuss where PHP can be extended and how it's done.

Capitolo 26. Extension Possibilities

As shown in Figura 25-1 above, PHP can be extended primarily at three points: external modules, built-in modules, and the Zend engine. The following sections discuss these options.

External Modules

External modules can be loaded at script runtime using the function dl(). This function loads a shared object from disk and makes its functionality available to the script to which it's being bound. After the script is terminated, the external module is discarded from memory. This method has both advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
External modules don't require recompiling of PHP.	The shared objects need to be loaded every time a script is being executed (every hit), which is very slow.
The size of PHP remains small by "outsourcing" certain functionality.	External additional files clutter up the disk.
	Every script that wants to use an external module's functionality has to specifically include a call to dl(), or the `extension` tag in `php.ini` needs to be modified (which is not always a suitable solution).

To sum up, external modules are great for third-party products, small additions to PHP that are rarely used, or just for testing purposes. To develop additional functionality quickly, external modules provide the best results. For frequent usage, larger implementations, and complex code, the disadvantages outweigh the advantages.

Third parties might consider using the `extension` tag in `php.ini` to create additional external modules to PHP. These external modules are completely detached from the main package, which is a very handy feature in commercial environments. Commercial distributors can simply ship disks or archives containing only their additional modules, without the need to create fixed and solid PHP binaries that don't allow other modules to be bound to them.

Built-in Modules

Built-in modules are compiled directly into PHP and carried around with every PHP process; their functionality is instantly available to every script that's being run. Like external modules, built-in modules have advantages and disadvantages, as described in the following table:

Advantages	Disadvantages
No need to load the module specifically; the functionality is instantly available.	Changes to built-in modules require recompiling of PHP.
No external files clutter up the disk; everything resides in the PHP binary.	The PHP binary grows and consumes more memory.

Built-in modules are best when you have a solid library of functions that remains relatively unchanged, requires better than poor-to-average performance, or is used frequently by many scripts

on your site. The need to recompile PHP is quickly compensated by the benefit in speed and ease of use. However, built-in modules are not ideal when rapid development of small additions is required.

The Zend Engine

Of course, extensions can also be implemented directly in the Zend engine. This strategy is good if you need a change in the language behavior or require special functions to be built directly into the language core. In general, however, modifications to the Zend engine should be avoided. Changes here result in incompatibilities with the rest of the world, and hardly anyone will ever adapt to specially patched Zend engines. Modifications can't be detached from the main PHP sources and are overridden with the next update using the "official" source repositories. Therefore, this method is generally considered bad practice and, due to its rarity, is not covered in this book.

Capitolo 27. Source Layout

> **Nota:** Prior to working through the rest of this chapter, you should retrieve clean, unmodified source trees of your favorite Web server. We're working with Apache (available at http://www.apache.org/) and, of course, with PHP (available at http://www.php.net/ - does it need to be said?).
>
> Make sure that you can compile a working PHP environment by yourself! We won't go into this issue here, however, as you should already have this most basic ability when studying this chapter.

Before we start discussing code issues, you should familiarize yourself with the source tree to be able to quickly navigate through PHP's files. This is a must-have ability to implement and debug extensions.

After extracting the PHP archive, you'll see a directory layout similar to that in Figura 27-1.

Figura 27-1. Main directory layout of the PHP source tree.

The following table describes the contents of the major directories.

Directory	Contents
`php-4`	Main PHP source files and main header files; here you'll find all of PHP's API definitions, macros, etc. (imp
`ext`	Repository for dynamic and built-in modules; by default, these are the "official" PHP modules that have bee
`pear`	Directory for the PHP class repository. At the time of this writing, this is still in the design phase, but it's bei
`sapi`	Contains the code for the different server abstraction layers.
`TSRM`	Location of the "Thread Safe Resource Manager" (TSRM) for Zend and PHP.
`Zend`	Location of Zend's file; here you'll find all of Zend's API definitions, macros, etc. (important).

Discussing all the files included in the PHP package is beyond the scope of this chapter. However, you should take a close look at the following files:

- `php.h`, located in the main PHP directory. This file contains most of PHP's macro and API definitions.
- `zend.h`, located in the main Zend directory. This file contains most of Zend's macros and definitions.
- `zend_API.h`, also located in the Zend directory, which defines Zend's API.

You should also follow some sub-inclusions from these files; for example, the ones relating to the Zend executor, the PHP initialization file support, and such. After reading these files, take the time to navigate around the package a little to see the interdependencies of all files and modules - how they

relate to each other and especially how they make use of each other. This also helps you to adapt to the coding style in which PHP is authored. To extend PHP, you should quickly adapt to this style.

Extension Conventions

Zend is built using certain conventions; to avoid breaking its standards, you should follow the rules described in the following sections.

Macros

For almost every important task, Zend ships predefined macros that are extremely handy. The tables and figures in the following sections describe most of the basic functions, structures, and macros. The macro definitions can be found mainly in `zend.h` and `zend_API.h`. We suggest that you take a close look at these files after having studied this chapter. (Although you can go ahead and read them now, not everything will make sense to you yet.)

Memory Management

Resource management is a crucial issue, especially in server software. One of the most valuable resources is memory, and memory management should be handled with extreme care. Memory management has been partially abstracted in Zend, and you should stick to this abstraction for obvious reasons: Due to the abstraction, Zend gets full control over all memory allocations. Zend is able to determine whether a block is in use, automatically freeing unused blocks and blocks with lost references, and thus prevent memory leaks. The functions to be used are described in the following table:

Function	Description
emalloc()	Serves as replacement for **malloc()**.
efree()	Serves as replacement for **free()**.
estrdup()	Serves as replacement for **strdup()**.
estrndup()	Serves as replacement for **strndup()**. Faster than **estrdup()** and binary-safe. This is the recommended funct
ecalloc()	Serves as replacement for **calloc()**.
erealloc()	Serves as replacement for **realloc()**.

emalloc(), **estrdup()**, **estrndup()**, **ecalloc()**, and **erealloc()** allocate internal memory; **efree()** frees these previously allocated blocks. Memory handled by the **e*()** functions is considered local to the current process and is discarded as soon as the script executed by this process is terminated.

Attenzione

To allocate resident memory that survives termination of the current script, you can use **malloc()** and **free()**. This should only be done with extreme care, however, and only in conjunction with demands of the Zend API; otherwise, you risk memory leaks.

Zend also features a thread-safe resource manager to provide better native support for multithreaded

Web servers. This requires you to allocate local structures for all of your global variables to allow concurrent threads to be run. Because the thread-safe mode of Zend was not finished back when this was written, it is not yet extensively covered here.

Directory and File Functions

The following directory and file functions should be used in Zend modules. They behave exactly like their C counterparts, but provide virtual working directory support on the thread level.

Zend Function	Regular C Function
V_GETCWD()	getcwd()
V_FOPEN()	fopen()
V_OPEN()	**open()**
V_CHDIR()	chdir()
V_GETWD()	**getwd()**
V_CHDIR_FILE()	Takes a file path as an argument and changes the current working directory to that file's directory.
V_STAT()	stat()
V_LSTAT()	lstat()

String Handling

Strings are handled a bit differently by the Zend engine than other values such as integers, Booleans, etc., which don't require additional memory allocation for storing their values. If you want to return a string from a function, introduce a new string variable to the symbol table, or do something similar, you have to make sure that the memory the string will be occupying has previously been allocated, using the aforementioned **e*()** functions for allocation. (This might not make much sense to you yet; just keep it somewhere in your head for now - we'll get back to it shortly.)

Complex Types

Complex types such as arrays and objects require different treatment. Zend features a single API for these types - they're stored using hash tables.

> **Nota:** To reduce complexity in the following source examples, we're only working with simple types such as integers at first. A discussion about creating more advanced types follows later in this chapter.

Capitolo 28. PHP’s Automatic Build System

PHP 4 features an automatic build system that's very flexible. All modules reside in a subdirectory of the `ext` directory. In addition to its own sources, each module consists of an M4 file (for example, see http://www.gnu.org/manual/m4/html_mono/m4.html) for configuration and a Makefile.in file, which is responsible for compilation (the results of autoconf and automake; for example, see http://sourceware.cygnus.com/autoconf/autoconf.html and http://sourceware.cygnus.com/automake/automake.html).

Both files are generated automatically, along with `.cvsignore`, by a little shell script named `ext_skel` that resides in the `ext` directory. As argument it takes the name of the module that you want to create. The shell script then creates a directory of the same name, along with the appropriate `config.m4` and `Makefile.in` files.

Step by step, the process looks like this:

```
root@dev:/usr/local/src/php4/ext > ./ext_skel my_module
Creating directory
Creating basic files: config.m4 Makefile.in .cvsignore [done].
To use your new extension, you will have to execute the following steps:
    $ cd ..
    $ ./buildconf
    $ ./configure # (your extension is automatically enabled)
    $ vi ext/my_module/my_module.c
    $ make
Repeat the last two steps as often as necessary.
```

This instruction creates the aforementioned files. To include the new module in the automatic configuration and build process, you have to run `buildconf`, which regenerates the `configure` script by searching through the `ext` directory and including all found `config.m4` files.

Finally, running `configure` parses all configuration options and generates a makefile based on those options and the options you specify in `Makefile.in`.

Esempio 28-1shows the previously generated `Makefile.in`:

Esempio 28-1. The default `Makefile.in`.

```
# $Id: Extending_Zend_Build.xml,v 1.6 2002/03/25 08:13:46 hholzgra Exp $
LTLIBRARY_NAME        = libmy_module.la
LTLIBRARY_SOURCES     = my_module.c
LTLIBRARY_SHARED_NAME = my_module.la include
$(top_srcdir)/build/dynlib.mk
```

There's not much to tell about this one: It contains the names of the input and output files. You could also specify build instructions for other files if your module is built from multiple source files.

The default `config.m4` shown in Esempio 28-2'/> is a bit more complex:

Esempio 28-2. The default `config.m4`.

```
dnl $Id: Extending_Zend_Build.xml,v 1.6 2002/03/25 08:13:46 hholzgra Exp $
dnl config.m4 for extension my_module
dnl don't forget to call PHP_EXTENSION(my_module)
dnl If your extension references something external, use with:
PHP_ARG_WITH(my_module, for my_module support,
dnl Make sure that the comment is aligned:
    [  --with-my_module             Include my_module support])
```

```
dnl Otherwise use enable:
PHP_ARG_ENABLE(my_module, whether to enable my_module support,
dnl Make sure that the comment is aligned:
    [  --enable-my_module           Enable my_module support])
if test "$PHP_MY_MODULE" != "no"; then
dnl Action..
PHP_EXTENSION(my_module, $ext_shared)
fi
```

If you're unfamiliar with M4 files (now is certainly a good time to get familiar), this might be a bit confusing at first; but it's actually quite easy.

Note: Everything prefixed with `dnl` is treated as a comment and is not parsed.

The `config.m4` file is responsible for parsing the command-line options passed to `configure` at configuration time. This means that it has to check for required external files and do similar configuration and setup tasks.

The default file creates two configuration directives in the `configure` script: `--with-my_module` and `--enable-my_module`. Use the first option when referring external files (such as the `--with-apache` directive that refers to the Apache directory). Use the second option when the user simply has to decide whether to enable your extension. Regardless of which option you use, you should uncomment the other, unnecessary one; that is, if you're using `--enable-my_module`, you should remove support for `--with-my_module`, and vice versa.

By default, the `config.m4` file created by `ext_skel` accepts both directives and automatically enables your extension. Enabling the extension is done by using the `PHP_EXTENSION` macro. To change the default behavior to include your module into the PHP binary when desired by the user (by explicitly specifying `--enable-my_module` or `--with-my_module`), change the test for `$PHP_MY_MODULE` to `== "yes"`:

```
if test "$PHP_MY_MODULE" == "yes"; then dnl
    Action.. PHP_EXTENSION(my_module, $ext_shared)
    fi
```

This would require you to use `--enable-my_module` each time when reconfiguring and recompiling PHP.

Note: Be sure to run `buildconf` every time you change `config.m4`!

We'll go into more details on the M4 macros available to your configuration scripts later in this chapter. For now, we'll simply use the default files. The sample sources on the CD-ROM all have working `config.m4` files. To include them into the PHP build process, simply copy the source directories to your PHP `ext` directory, run `buildconf`, and then include the sample modules you want by using the appropriate `--enable-*` directives with `configure`.

Capitolo 29. Creating Extensions

We'll start with the creation of a very simple extension at first, which basically does nothing more than implement a function that returns the integer it receives as parameter. Esempio 29-1 shows the source.

Esempio 29-1. A simple extension.

```
/* include standard header */
#include "php.h"

/* declaration of functions to be exported */
ZEND_FUNCTION(first_module);

/* compiled function list so Zend knows what's in this module */
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};

/* compiled module information */
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL,
    NULL,
    NULL,
    NULL,
    NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES
};

/* implement standard "stub" routine to introduce ourselves to Zend */
#if COMPILE_DL_FIRST_MODULE
ZEND_GET_MODULE(firstmod)
#endif

/* implement function that is meant to be made available to PHP */
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAIL-
URE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

This code contains a complete PHP module. We'll explain the source code in detail shortly, but first we'd like to discuss the build process. (This will allow the impatient to experiment before we dive into API discussions.)

Nota: The example source makes use of some features introduced with the Zend version used in PHP 4.1.0 and above, it won't compile with older PHP 4.0.x versions.

Compiling Modules

There are basically two ways to compile modules:

- Use the provided "make" mechanism in the `ext` directory, which also allows building of dynamic loadable modules.
- Compile the sources manually.

The first method should definitely be favored, since, as of PHP 4.0, this has been standardized into a sophisticated build process. The fact that it is so sophisticated is also its drawback, unfortunately - it's hard to understand at first. We'll provide a more detailed introduction to this later in the chapter, but first let's work with the default files.

The second method is good for those who (for some reason) don't have the full PHP source tree available, don't have access to all files, or just like to juggle with their keyboard. These cases should be extremely rare, but for the sake of completeness we'll also describe this method.

Compiling Using Make. To compile the sample sources using the standard mechanism, copy all their subdirectories to the `ext` directory of your PHP source tree. Then run `buildconf`, which will create an updated `configure` script containing appropriate options for the new extension. By default, all the sample sources are disabled, so you don't have to fear breaking your build process.

After you run `buildconf`, `configure --help` shows the following additional modules:

```
--enable-array_experiments   BOOK: Enables array experiments
--enable-call_userland       BOOK: Enables userland module
--enable-cross_conversion    BOOK: Enables cross-conversion module
--enable-first_module        BOOK: Enables first module
--enable-infoprint           BOOK: Enables infoprint module
--enable-reference_test      BOOK: Enables reference test module
--enable-resource_test       BOOK: Enables resource test module
--enable-variable_creation   BOOK: Enables variable-creation module
```

The module shown earlier in Esempio 29-1 can be enabled with `--enable-first_module` or `--enable-first_module=yes`.

Compiling Manually. To compile your modules manually, you need the following commands:

Action	Command
Compiling	cc -fpic -DCOMPILE_DL=1 -I/usr/local/include -I. -I.. -I../Zend -c -o `<your_object_file>` `<your_c_fi`
Linking	cc -shared -L/usr/local/lib -rdynamic -o `<your_module_file>` `<your_object_file(s)>`

The command to compile the module simply instructs the compiler to generate position-independent code (`-fpic` shouldn't be omitted) and additionally defines the constant `COMPILE_DL` to tell the module code that it's compiled as a dynamically loadable module (the test module above checks for this; we'll discuss it shortly). After these options, it specifies a number of standard include paths that should be used as the minimal set to compile the source files.

Note: All include paths in the example are relative to the directory `ext`. If you're compiling from another directory, change the pathnames accordingly. Required items are the PHP directory, the `Zend` directory, and (if necessary), the directory in which your module resides.

The link command is also a plain vanilla command instructing linkage as a dynamic module.

You can include optimization options in the compilation command, although these have been omitted in this example (but some are included in the makefile template described in an earlier section).

Note: Compiling and linking manually as a static module into the PHP binary involves very long instructions and thus is not discussed here. (It's not very efficient to type all those commands.)

Capitolo 30. Using Extensions

Depending on the build process you selected, you should either end up with a new PHP binary to be linked into your Web server (or run as CGI), or with an .so (shared object) file. If you compiled the example file `first_module.c` as a shared object, your result file should be `first_module.so`. To use it, you first have to copy it to a place from which it's accessible to PHP. For a simple test procedure, you can copy it to your `htdocs` directory and try it with the source in Esempio 30-1. If you compiled it into the PHP binary, omit the call to dl(), as the module's functionality is instantly available to your scripts.

> **Attenzione**
>
> For security reasons, you *should not* put your dynamic modules into publicly accessible directories. Even though it *can* be done and it simplifies testing, you should put them into a separate directory in production environments.

Esempio 30-1. A test file for first_module.so.

```
<?php

// remove next comment if necessary
// dl("first_module.so");

$param = 2;
$return = first_module($param);

print("We sent '$param' and got '$return'");

?>
```

Calling this PHP file in your Web browser should give you the output shown in Figura 30-1.

Figura 30-1. Output of first_module.php.

If required, the dynamic loadable module is loaded by calling the dl() function. This function looks for the specified shared object, loads it, and makes its functions available to PHP. The module exports the function **first_module()**, which accepts a single parameter, converts it to an integer, and returns the result of the conversion.

If you've gotten this far, congratulations! You just built your first extension to PHP.

Capitolo 31. Troubleshooting

Actually, not much troubleshooting can be done when compiling static or dynamic modules. The only problem that could arise is that the compiler will complain about missing definitions or something similar. In this case, make sure that all header files are available and that you specified their path correctly in the compilation command. To be sure that everything is located correctly, extract a clean PHP source tree and use the automatic build in the `ext` directory with the fresh files; this will guarantee a safe compilation environment. If this fails, try manual compilation.

PHP might also complain about missing functions in your module. (This shouldn't happen with the sample sources if you didn't modify them.) If the names of external functions you're trying to access from your module are misspelled, they'll remain as "unlinked symbols" in the symbol table. During dynamic loading and linkage by PHP, they won't resolve because of the typing errors - there are no corresponding symbols in the main binary. Look for incorrect declarations in your module file or incorrectly written external references. Note that this problem is specific to dynamic loadable modules; it doesn't occur with static modules. Errors in static modules show up at compile time.

Capitolo 32. Source Discussion

Now that you've got a safe build environment and you're able to include the modules into PHP files, it's time to discuss how everything works.

Module Structure

All PHP modules follow a common structure:

- Header file inclusions (to include all required macros, API definitions, etc.)
- C declaration of exported functions (required to declare the Zend function block)
- Declaration of the Zend function block
- Declaration of the Zend module block
- Implementation of **get_module()**
- Implementation of all exported functions

Header File Inclusions

The only header file you really have to include for your modules is `php.h`, located in the PHP directory. This file makes all macros and API definitions required to build new modules available to your code.

Tip: It's good practice to create a separate header file for your module that contains module-specific definitions. This header file should contain all the forward definitions for exported functions and include `php.h`. If you created your module using `ext_skel` you already have such a header file prepared.

Declaring Exported Functions

To declare functions that are to be exported (i.e., made available to PHP as new native functions), Zend provides a set of macros. A sample declaration looks like this:

```
ZEND_FUNCTION ( my_function );
```

`ZEND_FUNCTION` declares a new C function that complies with Zend's internal API. This means that the function is of type `void` and accepts `INTERNAL_FUNCTION_PARAMETERS` (another macro) as parameters. Additionally, it prefixes the function name with `zif`. The immediately expanded version of the above definitions would look like this:

```
void zif_my_function ( INTERNAL_FUNCTION_PARAMETERS );
```

Expanding `INTERNAL_FUNCTION_PARAMETERS` results in the following:

```
void zif_my_function( int ht
```

```
, zval * return_value
, zval * this_ptr
, int return_value_used
, zend_executor_globals * executor_globals
);
```

Since the interpreter and executor core have been separated from the main PHP package, a second API defining macros and function sets has evolved: the Zend API. As the Zend API now handles quite a few of the responsibilities that previously belonged to PHP, a lot of PHP functions have been reduced to macros aliasing to calls into the Zend API. The recommended practice is to use the Zend API wherever possible, as the old API is only preserved for compatibility reasons. For example, the types zval and pval are identical. zval is Zend's definition; pval is PHP's definition (actually, pval is an alias for zval now). As the macro `INTERNAL_FUNCTION_PARAMETERS` is a Zend macro, the above declaration contains zval. When writing code, you should always use zval to conform to the new Zend API.

The parameter list of this declaration is very important; you should keep these parameters in mind (see Tabella 32-1 for descriptions).

Tabella 32-1. Zend's Parameters to Functions Called from PHP

Parameter	Description
ht	The number of arguments passed to the Zend function. You should not touch this directly, but instea
return_value	This variable is used to pass any return values of your function back to PHP. Access to this variable
this_ptr	Using this variable, you can gain access to the object in which your function is contained, if it's use
return_value_used	This flag indicates whether an eventual return value from this function will actually be used by the c
executor_globals	This variable points to global settings of the Zend engine. You'll find this useful when creating new

Declaration of the Zend Function Block

Now that you have declared the functions to be exported, you also have to introduce them to Zend. Introducing the list of functions is done by using an array of zend_function_entry. This array consecutively contains all functions that are to be made available externally, with the function's name as it should appear in PHP and its name as defined in the C source. Internally, zend_function_entry is defined as shown in Esempio 32-1.

Esempio 32-1. Internal declaration of zend_function_entry.

```
typedef struct _zend_function_entry {
    char *fname;
    void (*handler)(INTERNAL_FUNCTION_PARAMETERS);
    unsigned char *func_arg_types;
} zend_function_entry;
```

Entry	Description

fname	Denotes the function name as seen in PHP (for example, `fopen`, `mysql_connect`, or, in our example,
handler	Pointer to the C function responsible for handling calls to this function. For example, see the standard
func_arg_types	Allows you to mark certain parameters so that they're forced to be passed by reference. You usually sh

In the example above, the declaration looks like this:

```
zend_function_entry firstmod_functions[] =
{
    ZEND_FE(first_module, NULL)
    {NULL, NULL, NULL}
};
```

You can see that the last entry in the list always has to be `{NULL, NULL, NULL}`. This marker has to be set for Zend to know when the end of the list of exported functions is reached.

> **Nota:** You *cannot* use the predefined macros for the end marker, as these would try to refer to a function named "NULL"!

The macro `ZEND_FE` (short for 'Zend Function Entry') simply expands to a structure entry in zend_function_entry. Note that these macros introduce a special naming scheme to your functions - your C functions will be prefixed with `zif_`, meaning that `ZEND_FE(first_module)` will refer to a C function **zif_first_module()**. If you want to mix macro usage with hand-coded entries (not a good practice), keep this in mind.

Tip: Compilation errors that refer to functions named **zif_*()** relate to functions defined with `ZEND_FE`.

Tabella 32-2 shows a list of all the macros that you can use to define functions.

Tabella 32-2. Macros for Defining Functions

Macro Name	Description
`ZEND_FE(name, arg_types)`	Defines a function entry of the name name in zend_functio
`ZEND_NAMED_FE(php_name, name, arg_types)`	Defines a function that will be available to PHP by the nam
`ZEND_FALIAS(name, alias, arg_types)`	Defines an alias named alias for name. arg_types needs to
`PHP_FE(name, arg_types)`	Old PHP API equivalent of `ZEND_FE`.
`PHP_NAMED_FE(runtime_name, name, arg_types)`	Old PHP API equivalent of `ZEND_NAMED_FE`.

Note: You can't use `ZEND_FE` in conjunction with `PHP_FUNCTION`, or `PHP_FE` in conjunction with `ZEND_FUNCTION`. However, it's perfectly legal to mix `ZEND_FE` and `ZEND_FUNCTION` with `PHP_FE` and `PHP_FUNCTION` when staying with the same macro set for each function to be declared. But mixing is *not* recommended; instead, you're advised to use the `ZEND_*` macros only.

Declaration of the Zend Module Block

This block is stored in the structure zend_module_entry and contains all necessary information to

describe the contents of this module to Zend. You can see the internal definition of this module in Esempio 32-2.

Esempio 32-2. Internal declaration of zend_module_entry.

```
typedef struct _zend_module_entry zend_module_entry;

    struct _zend_module_entry {
    unsigned short size;
    unsigned int zend_api;
    unsigned char zend_debug;
    unsigned char zts;
    char *name;
    zend_function_entry *functions;
    int (*module_startup_func)(INIT_FUNC_ARGS);
    int (*module_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    int (*request_startup_func)(INIT_FUNC_ARGS);
    int (*request_shutdown_func)(SHUTDOWN_FUNC_ARGS);
    void (*info_func)(ZEND_MODULE_INFO_FUNC_ARGS);
    char *version;
    int (*global_startup_func)(void);
    int (*global_shutdown_func)(void);

[ Rest of the structure is not interesting here ]

};
```

Entry	Description
size, zend_api, zend_debug and zts	Usually filled with the "STANDARD_MODULE_HEADER", which fills these four mer
name	Contains the module name (for example, "File functions", "Socket funct
functions	Points to the Zend function block, discussed in the preceding section.
module_startup_func	This function is called once upon module initialization and can be used to do one-
module_shutdown_func	This function is called once upon module shutdown and can be used to do one-tim
request_startup_func	This function is called once upon every page request and can be used to do one-ti
request_shutdown_func	This function is called once after every page request and works as counterpart to r
info_func	When phpinfo() is called in a script, Zend cycles through all loaded modules and
version	The version of the module. You can use NO_VERSION_YET if you don't want to g
Remaining structure elements	These are used internally and can be prefilled by using the macro STANDARD_MOD

In our example, this structure is implemented as follows:

```
zend_module_entry firstmod_module_entry =
{
    STANDARD_MODULE_HEADER,
    "First Module",
    firstmod_functions,
    NULL, NULL, NULL, NULL, NULL,
    NO_VERSION_YET,
    STANDARD_MODULE_PROPERTIES,
};
```

This is basically the easiest and most minimal set of values you could ever use. The module name is set to `First Module`, then the function list is referenced, after which all startup and shutdown functions are marked as being unused.

For reference purposes, you can find a list of the macros involved in declared startup and shutdown functions in Tabella 32-3. These are not used in our basic example yet, but will be demonstrated later on. You should make use of these macros to declare your startup and shutdown functions, as these require special arguments to be passed (`INIT_FUNC_ARGS` and `SHUTDOWN_FUNC_ARGS`), which are automatically included into the function declaration when using the predefined macros. If you declare your functions manually and the PHP developers decide that a change in the argument list is necessary, you'll have to change your module sources to remain compatible.

Tabella 32-3. Macros to Declare Startup and Shutdown Functions

Macro	Description
`ZEND_MINIT(module)`	Declares a function for module startup. The generated name will be `zend_minit_<mod`
`ZEND_MSHUTDOWN(module)`	Declares a function for module shutdown. The generated name will be `zend_mshutdow`
`ZEND_RINIT(module)`	Declares a function for request startup. The generated name will be `zend_rinit_<mod`
`ZEND_RSHUTDOWN(module)`	Declares a function for request shutdown. The generated name will be `zend_rshutdow`
`ZEND_MINFO(module)`	Declares a function for printing module information, used when phpinfo() is called. The

Creation of get_module()

This function is special to all dynamic loadable modules. Take a look at the creation via the `ZEND_GET_MODULE` macro first:

```
#if COMPILE_DL_FIRSTMOD
     ZEND_GET_MODULE(firstmod)
#endif
```

The function implementation is surrounded by a conditional compilation statement. This is needed since the function **get_module()** is only required if your module is built as a dynamic extension. By specifying a definition of `COMPILE_DL_FIRSTMOD` in the compiler command (see above for a discussion of the compilation instructions required to build a dynamic extension), you can instruct your module whether you intend to build it as a dynamic extension or as a built-in module. If you want a built-in module, the implementation of **get_module()** is simply left out.

get_module() is called by Zend at load time of the module. You can think of it as being invoked by the dl() call in your script. Its purpose is to pass the module information block back to Zend in order to inform the engine about the module contents.

If you don't implement a **get_module()** function in your dynamic loadable module, Zend will compliment you with an error message when trying to access it.

Implementation of All Exported Functions

Implementing the exported functions is the final step. The example function in `first_module`

looks like this:

```
ZEND_FUNCTION(first_module)
{
    long parameter;

    if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "l", &parameter) == FAIL-
URE) {
        return;
    }

    RETURN_LONG(parameter);
}
```

The function declaration is done using `ZEND_FUNCTION`, which corresponds to `ZEND_FE` in the function entry table (discussed earlier).

After the declaration, code for checking and retrieving the function's arguments, argument conversion, and return value generation follows (more on this later).

Summary

That's it, basically - there's nothing more to implementing PHP modules. Built-in modules are structured similarly to dynamic modules, so, equipped with the information presented in the previous sections, you'll be able to fight the odds when encountering PHP module source files.

Now, in the following sections, read on about how to make use of PHP's internals to build powerful extensions.

Capitolo 33. Accepting Arguments

One of the most important issues for language extensions is accepting and dealing with data passed via arguments. Most extensions are built to deal with specific input data (or require parameters to perform their specific actions), and function arguments are the only real way to exchange data between the PHP level and the C level. Of course, there's also the possibility of exchanging data using predefined global values (which is also discussed later), but this should be avoided by all means, as it's extremely bad practice.

PHP doesn't make use of any formal function declarations; this is why call syntax is always completely dynamic and never checked for errors. Checking for correct call syntax is left to the user code. For example, it's possible to call a function using only one argument at one time and four arguments the next time - both invocations are syntactically absolutely correct.

Determining the Number of Arguments

Since PHP doesn't have formal function definitions with support for call syntax checking, and since PHP features variable arguments, sometimes you need to find out with how many arguments your function has been called. You can use the `ZEND_NUM_ARGS` macro in this case. In previous versions of PHP, this macro retrieved the number of arguments with which the function has been called based on the function's hash table entry, ht, which is passed in the `INTERNAL_FUNCTION_PARAMETERS` list. As ht itself now contains the number of arguments that have been passed to the function, `ZEND_NUM_ARGS` has been stripped down to a dummy macro (see its definition in `zend_API.h`). But it's still good practice to use it, to remain compatible with future changes in the call interface. *Note:* The old PHP equivalent of this macro is `ARG_COUNT`.

The following code checks for the correct number of arguments:

```
if(ZEND_NUM_ARGS() != 2) WRONG_PARAM_COUNT;
```

If the function is not called with two arguments, it exits with an error message. The code snippet above makes use of the tool macro `WRONG_PARAM_COUNT`, which can be used to generate a standard error message (see Figura 33-1).

Figura 33-1. `WRONG_PARAM_COUNT` in action.

This macro prints a default error message and then returns to the caller. Its definition can also be found in `zend_API.h` and looks like this:

```
ZEND_API void wrong_param_count(void);

#define WRONG_PARAM_COUNT { wrong_param_count(); return; }
```

As you can see, it calls an internal function named **wrong_param_count()** that's responsible for printing the warning. For details on generating customized error messages, see the later section "Printing Information."

Retrieving Arguments

> **New parameter parsing API:** This chapter documents the new Zend parameter parsing API introduced by Andrei Zmievski. It was introduced in the development stage between PHP 4.0.6 and 4.1.0 .

Parsing parameters is a very common operation and it may get a bit tedious. It would also be nice to have standardized error checking and error messages. Since PHP 4.1.0, there is a way to do just that by using the new parameter parsing API. It greatly simplifies the process of receiving parameteres, but it has a drawback in that it can't be used for functions that expect variable number of parameters. But since the vast majority of functions do not fall into those categories, this parsing API is recommended as the new standard way.

The prototype for parameter parsing function looks like this:

```
int zend_parse_parameters(int num_args TSRMLS_DC, char *type_spec, ...);
```

The first argument to this function is supposed to be the number of actual parameters passed to your function, so `ZEND_NUM_ARGS()` can be used for that. The second parameter should always be `TSRMLS_CC` macro. The third argument is a string that specifies the number and types of arguments your function is expecting, similar to how printf format string specifies the number and format of the output values it should operate on. And finally the rest of the arguments are pointers to variables which should receive the values from the parameters.

zend_parse_parameters() also performs type conversions whenever possible, so that you always receive the data in the format you asked for. Any type of scalar can be converted to another one, but conversions between complex types (arrays, objects, and resources) and scalar types are not allowed.

If the parameters could be obtained successfully and there were no errors during type conversion, the function will return `SUCCESS`, otherwise it will return `FAILURE`. The function will output informative error messages, if the number of received parameters does not match the requested number, or if type conversion could not be performed.

Here are some sample error messages:

```
Warning - ini_get_all() requires at most 1 parameter, 2 given
Warning - wddx_deserialize() expects parameter 1 to be string, array given
```

Of course each error message is accompanied by the filename and line number on which it occurs.

Here is the full list of type specifiers:

- `l` - long
- `d` - double
- `s` - string (with possible null bytes) and its length
- `b` - boolean

- `r` - resource, stored in `zval*`
- `a` - array, stored in `zval*`
- `o` - object (of any class), stored in `zval*`
- `O` - object (of class specified by class entry), stored in `zval*`
- `z` - the actual `zval*`

The following characters also have a meaning in the specifier string:

- `|` - indicates that the remaining parameters are optional. The storage variables corresponding to these parameters should be initialized to default values by the extension, since they will not be touched by the parsing function if the parameters are not passed.
- `/` - the parsing function will call **SEPARATE_ZVAL_IF_NOT_REF()** on the parameter it follows, to provide a copy of the parameter, unless it's a reference.
- `!` - the parameter it follows can be of specified type or `NULL` (only applies to a, o, O, r, and z). If `NULL` value is passed by the user, the storage pointer will be set to `NULL`.

The best way to illustrate the usage of this function is through examples:

```
/* Gets a long, a string and its length, and a zval. */
long l;
char *s;
int s_len;
zval *param;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "lsz", &l, &s, &s_len, &param) == FAILURE) {
    return;
}

/* Gets an object of class specified by my_ce, and an optional double. */
zval *obj;
double d = 0.5;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC,
                          "O|d", &obj, my_ce, &d) == FAILURE) {
    return;
}

/* Gets an object or null, and an array.
   If null is passed for object, obj will be set to NULL. */
zval *obj;
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "O!a", &obj, &arr) == FAIL-
URE) {
    return;
}

/* Gets a separated array. */
zval *arr;
if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "a/", &arr) == FAILURE) {
    return;
}

/* Get only the first three parameters (useful for varargs functions). */
```

```
zval *z;
zend_bool b;
zval *r;
if (zend_parse_parameters(3, "zbr!", &z, &b, &r) == FAILURE) {
    return;
}
```

Note that in the last example we pass 3 for the number of received parameters, instead of **ZEND_NUM_ARGS()**. What this lets us do is receive the least number of parameters if our function expects a variable number of them. Of course, if you want to operate on the rest of the parameters, you will have to use **zend_get_parameters_array_ex()** to obtain them.

The parsing function has an extended version that allows for an additional flags argument that controls its actions.

```
int zend_parse_parameters_ex(int flags, int num_args TSRMLS_DC, char *type_spec, ...);
```

The only flag you can pass currently is `ZEND_PARSE_PARAMS_QUIET`, which instructs the function to not output any error messages during its operation. This is useful for functions that expect several sets of completely different arguments, but you will have to output your own error messages.

For example, here is how you would get either a set of three longs or a string:

```
long l1, l2, l3;
char *s;
if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                             ZEND_NUM_ARGS() TSRMLS_CC,
                             "lll", &l1, &l2, &l3) == SUCCESS) {
    /* manipulate longs */
} else if (zend_parse_parameters_ex(ZEND_PARSE_PARAMS_QUIET,
                                    ZEND_NUM_ARGS(), "s", &s, &s_len) == SUC-
CESS) {
    /* manipulate string */
} else {
    php_error(E_WARNING, "%s() takes either three long values or a string as argument",
              get_active_function_name(TSRMLS_C));
    return;
}
```

With all the abovementioned ways of receiving function parameters you should have a good handle on this process. For even more example, look through the source code for extensions that are shipped with PHP - they illustrate every conceivable situation.

Old way of retrieving arguments (deprecated)

> **Deprecated parameter parsing API:** This API is deprecated and superseded by the new ZEND parameter parsing API.

After having checked the number of arguments, you need to get access to the arguments themselves. This is done with the help of **zend_get_parameters_ex()**:

```
zval **parameter;

if(zend_get_parameters_ex(1, &parameter) != SUCCESS)
  WRONG_PARAM_COUNT;
```

All arguments are stored in a zval container, which needs to be pointed to *twice*. The snippet above tries to retrieve one argument and make it available to us via the parameter pointer.

zend_get_parameters_ex() accepts at least two arguments. The first argument is the number of arguments to retrieve (which should match the number of arguments with which the function has been called; this is why it's important to check for correct call syntax). The second argument (and all following arguments) are pointers to pointers to pointers to zvals. (Confusing, isn't it?) All these pointers are required because Zend works internally with **zval; to adjust a local **zval in our function,**zend_get_parameters_ex()** requires a pointer to it.

The return value of **zend_get_parameters_ex()** can either be `SUCCESS` or `FAILURE`, indicating (unsurprisingly) success or failure of the argument processing. A failure is most likely related to an incorrect number of arguments being specified, in which case you should exit with `WRONG_PARAM_COUNT`.

To retrieve more than one argument, you can use a similar snippet:

```
zval **param1, **param2, **param3, **param4;

if(zend_get_parameters_ex(4, &param1, &param2, &param3, &param4) != SUCCESS)
    WRONG_PARAM_COUNT;
```

zend_get_parameters_ex() only checks whether you're trying to retrieve too many parameters. If the function is called with five arguments, but you're only retrieving three of them with **zend_get_parameters_ex()**, you won't get an error but will get the first three parameters instead. Subsequent calls of **zend_get_parameters_ex()** won't retrieve the remaining arguments, but will get the same arguments again.

Dealing with a Variable Number of Arguments/Optional Parameters

If your function is meant to accept a variable number of arguments, the snippets just described are sometimes suboptimal solutions. You have to create a line calling **zend_get_parameters_ex()** for every possible number of arguments, which is often unsatisfying.

For this case, you can use the function **zend_get_parameters_array_ex()**, which accepts the number of parameters to retrieve and an array in which to store them:

```
zval **parameter_array[4];

/* get the number of arguments */
argument_count = ZEND_NUM_ARGS();

/* see if it satisfies our minimal request (2 arguments) */
/* and our maximal acceptance (4 arguments) */
if(argument_count < 2 || argument_count > 5)
    WRONG_PARAM_COUNT;

/* argument count is correct, now retrieve arguments */
if(zend_get_parameters_array_ex(argument_count, parameter_array) != SUCCESS)
    WRONG_PARAM_COUNT;
```

First, the number of arguments is checked to make sure that it's in the accepted range. After that, **zend_get_parameters_array_ex()** is used to fill parameter_array with valid pointers to the argument values.

A very clever implementation of this can be found in the code handling PHP's fsockopen() located in `ext/standard/fsock.c`, as shown in Esempio 33-1. Don't worry if you don't know all the functions used in this source yet; we'll get to them shortly.

Esempio 33-1. PHP's implementation of variable arguments in fsockopen().

```
pval **args[5];
int *sock=emalloc(sizeof(int));
int *sockp;
int arg_count=ARG_COUNT(ht);
int socketd = -1;
unsigned char udp = 0;
struct timeval timeout = { 60, 0 };
unsigned short portno;
unsigned long conv;
char *key = NULL;
FLS_FETCH();

if (arg_count > 5 || arg_count < 2 || zend_get_parameters_array_ex(arg_count,args)==FAILU
    CLOSE_SOCK(1);
    WRONG_PARAM_COUNT;
}

switch(arg_count) {
```

```
    case 5:
        convert_to_double_ex(args[4]);
        conv = (unsigned long) (Z_DVAL_P(args[4]) * 1000000.0);
        timeout.tv_sec = conv / 1000000;
        timeout.tv_usec = conv % 1000000;
        /* fall-through */
    case 4:
        if (!PZVAL_IS_REF(*args[3])) {
            php_error(E_WARNING,"error string argument to fsockopen not passed by referen
        }
        pval_copy_constructor(*args[3]);
        ZVAL_EMPTY_STRING(*args[3]);
        /* fall-through */
    case 3:
        if (!PZVAL_IS_REF(*args[2])) {
            php_error(E_WARNING,"error argument to fsockopen not passed by reference");
            return;
        }
        ZVAL_LONG(*args[2], 0);
        break;
}

convert_to_string_ex(args[0]);
convert_to_long_ex(args[1]);
portno = (unsigned short) Z_LVAL_P(args[1]);

key = emalloc(Z_STRLEN_P(args[0]) + 10);
```

fsockopen() accepts two, three, four, or five parameters. After the obligatory variable declarations, the function checks for the correct range of arguments. Then it uses a fall-through mechanism in a `switch()` statement to deal with all arguments. The `switch()` statement starts with the maximum number of arguments being passed (five). After that, it automatically processes the case of four arguments being passed, then three, by omitting the otherwise obligatory `break` keyword in all stages. After having processed the last case, it exits the `switch()` statement and does the minimal argument processing needed if the function is invoked with only two arguments.

This multiple-stage type of processing, similar to a stairway, allows convenient processing of a variable number of arguments.

Accessing Arguments

To access arguments, it's necessary for each argument to have a clearly defined type. Again, PHP's extremely dynamic nature introduces some quirks. Because PHP never does any kind of type checking, it's possible for a caller to pass any kind of data to your functions, whether you want it or not. If you expect an integer, for example, the caller might pass an array, and vice versa - PHP simply won't notice.

To work around this, you have to use a set of API functions to force a type conversion on every argument that's being passed (see Tabella 33-1).

Note: All conversion functions expect a **zval as parameter.

Tabella 33-1. Argument Conversion Functions

Function	Description
convert_to_boolean_ex()	Forces conversion to a Boolean type. Boolean values remain untouched. Longs, d
convert_to_long_ex()	Forces conversion to a long, the default integer type. NULL values, Booleans, res
convert_to_double_ex()	Forces conversion to a double, the default floating-point type. NULL values, Bool
convert_to_string_ex()	Forces conversion to a string. Strings remain untouched. NULL values are conver
`convert_to_array_ex(value)`	Forces conversion to an array. Arrays remain untouched. Objects are converted to
`convert_to_object_ex(value)`	Forces conversion to an object. Objects remain untouched. NULL values are conv
`convert_to_null_ex(value)`	Forces the type to become a NULL value, meaning empty.

> **Nota:** You can find a demonstration of the behavior in `cross_conversion.php` on the accompanying CD-ROM. Figura 33-2 shows the output.

Figura 33-2. Cross-conversion behavior of PHP.

Using these functions on your arguments will ensure type safety for all data that’s passed to you. If the supplied type doesn’t match the required type, PHP forces dummy contents on the resulting value (empty strings, arrays, or objects, `0` for numeric values, `FALSE` for Booleans) to ensure a defined state.

Following is a quote from the sample module discussed previously, which makes use of the conversion functions:

```
zval **parameter;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &parameter) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

convert_to_long_ex(parameter);

RETURN_LONG(Z_LVAL_P(parameter));
```

After retrieving the parameter pointer, the parameter value is converted to a long (an integer), which also forms the return value of this function. Understanding access to the contents of the value requires a short discussion of the zval type, whose definition is shown in Esempio 33-2.

Esempio 33-2. PHP/Zend zval type definition.

```
typedef pval zval;

typedef struct _zval_struct zval;

typedef union _zvalue_value {
long lval; /* long value */
double dval; /* double value */
struct {
char *val;
int len;
} str;
HashTable *ht; /* hash table value */
struct {
zend_class_entry *ce;
HashTable *properties;
} obj;
} zvalue_value;

struct _zval_struct {
/* Variable information */
zvalue_value value; /* value */
unsigned char type; /* active type */
unsigned char is_ref;
short refcount;
};
```

Actually, pval (defined in `php.h`) is only an alias of zval (defined in `zend.h`), which in turn refers to _zval_struct. This is a most interesting structure. _zval_struct is the "master" structure, containing the value structure, type, and reference information. The substructure zvalue_value is a union that contains the variable's contents. Depending on the variable's type, you'll have to access different members of this union. For a description of both structures, see Tabella 33-2, Tabella 33-3 and Tabella 33-4.

Tabella 33-2. Zend zval Structure

Entry	Description
value	Union containing this variable's contents. See Tabella 33-3 for a description.
type	Contains this variable's type. For a list of available types, see Tabella 33-4.
is_ref	0 means that this variable is not a reference; 1 means that this variable is a reference to another variable.
refcount	The number of references that exist for this variable. For every new reference to the value stored in this variab

Tabella 33-3. Zend zvalue_value Structure

Entry	Description
lval	Use this property if the variable is of the type `IS_LONG`, `IS_BOOLEAN`, or `IS_RESOURCE`.
dval	Use this property if the variable is of the type `IS_DOUBLE`.
str	This structure can be used to access variables of the type `IS_STRING`. The member len contains the string length

ht	This entry points to the variable's hash table entry if the variable is an array.
obj	Use this property if the variable is of the type `IS_OBJECT`.

Tabella 33-4. Zend Variable Type Constants

Constant	Description
`IS_NULL`	Denotes a NULL (empty) value.
`IS_LONG`	A long (integer) value.
`IS_DOUBLE`	A double (floating point) value.
`IS_STRING`	A string.
`IS_ARRAY`	Denotes an array.
`IS_OBJECT`	An object.
`IS_BOOL`	A Boolean value.
`IS_RESOURCE`	A resource (for a discussion of resources, see the appropriate section below).
`IS_CONSTANT`	A constant (defined) value.

To access a long you access zval.value.lval, to access a double you use zval.value.dval, and so on. Because all values are stored in a union, trying to access data with incorrect union members results in meaningless output.

Accessing arrays and objects is a bit more complicated and is discussed later.

Dealing with Arguments Passed by Reference

If your function accepts arguments passed by reference that you intend to modify, you need to take some precautions.

What we didn't say yet is that under the circumstances presented so far, you don't have write access to any zval containers designating function parameters that have been passed to you. Of course, you can change any zval containers that you created within your function, but you mustn't change any zvals that refer to Zend-internal data!

We've only discussed the so-called ***_ex()** API so far. You may have noticed that the API functions we've used are called **zend_get_parameters_ex()** instead of **zend_get_parameters()**, **convert_to_long_ex()** instead of **convert_to_long()**, etc. The ***_ex()** functions form the so-called new "extended" Zend API. They give a minor speed increase over the old API, but as a tradeoff are only meant for providing read-only access.

Because Zend works internally with references, different variables may reference the same value. Write access to a zval container requires this container to contain an isolated value, meaning a value that's not referenced by any other containers. If a zval container were referenced by other containers and you changed the referenced zval, you would automatically change the contents of the other containers referencing this zval (because they'd simply point to the changed value and thus change their own value as well).

zend_get_parameters_ex() doesn't care about this situation, but simply returns a pointer to the desired zval containers, whether they consist of references or not. Its corresponding function in the traditional API, **zend_get_parameters()**, immediately checks for referenced values. If it finds a

reference, it creates a new, isolated zval container; copies the referenced data into this newly allocated space; and then returns a pointer to the new, isolated value.

This action is called *zval separation* (or pval separation). Because the ***_ex()** API doesn't perform zval separation, it's considerably faster, while at the same time disabling write access.

To change parameters, however, write access is required. Zend deals with this situation in a special way: Whenever a parameter to a function is passed by reference, it performs automatic zval separation. This means that whenever you're calling a function like this in PHP, Zend will automatically ensure that $parameter is being passed as an isolated value, rendering it to a write-safe state:

```
my_function(&$parameter);
```

But this *is not* the case with regular parameters! All other parameters that are not passed by reference are in a read-only state.

This requires you to make sure that you're really working with a reference - otherwise you might produce unwanted results. To check for a parameter being passed by reference, you can use the macro `PZVAL_IS_REF`. This macro accepts a `zval*` to check if it is a reference or not. Examples are given in in Esempio 33-3.

Esempio 33-3. Testing for referenced parameter passing.

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
    return;

/* check for parameter being passed by reference */
if (!PZVAL_IS_REF(*parameter)) {
{
    zend_error(E_WARNING, "Parameter wasn't passed by reference");
    RETURN_NULL();
}

/* make changes to the parameter */
ZVAL_LONG(*parameter, 10);
```

Assuring Write Safety for Other Parameters

You might run into a situation in which you need write access to a parameter that's retrieved with **zend_get_parameters_ex()** but not passed by reference. For this case, you can use the macro SEPARATE_ZVAL, which does a zval separation on the provided container. The newly generated zval is detached from internal data and has only a local scope, meaning that it can be changed or destroyed without implying global changes in the script context:

```
zval **parameter;

/* retrieve parameter */
zend_get_parameters_ex(1, &parameter);

/* at this stage, <parameter> still is connected */
/* to Zend's internal data buffers */

/* make <parameter> write-safe */
SEPARATE_ZVAL(parameter);

/* now we can safely modify <parameter> */
/* without implying global changes */
```

SEPARATE_ZVAL uses **emalloc()** to allocate the new zval container, which means that even if you don't deallocate this memory yourself, it will be destroyed automatically upon script termination. However, doing a lot of calls to this macro without freeing the resulting containers will clutter up your RAM.

Note: As you can easily work around the lack of write access in the "traditional" API (with **zend_get_parameters()** and so on), this API seems to be obsolete, and is not discussed further in this chapter.

Capitolo 34. Creating Variables

When exchanging data from your own extensions with PHP scripts, one of the most important issues is the creation of variables. This section shows you how to deal with the variable types that PHP supports.

Overview

To create new variables that can be seen "from the outside" by the executing script, you need to allocate a new zval container, fill this container with meaningful values, and then introduce it to Zend's internal symbol table. This basic process is common to all variable creations:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol ta-
ble */
ZEND_SET_SYMBOL(EG(active_symbol_table), "new_variable_name", new_variable);

/* the variable is now accessible to the script by using $new_variable_name */
```

The macro `MAKE_STD_ZVAL` allocates a new zval container using `ALLOC_ZVAL` and initializes it using `INIT_ZVAL`. As implemented in Zend at the time of this writing, *initializing* means setting the reference count to `1` and clearing the is_ref flag, but this process could be extended later - this is why it's a good idea to keep using `MAKE_STD_ZVAL` instead of only using `ALLOC_ZVAL`. If you want to optimize for speed (and you don't have to explicitly initialize the zval container here), you can use `ALLOC_ZVAL`, but this isn't recommended because it doesn't ensure data integrity.

`ZEND_SET_SYMBOL` takes care of introducing the new variable to Zend's symbol table. This macro checks whether the value already exists in the symbol table and converts the new symbol to a reference if so (with automatic deallocation of the old zval container). This is the preferred method if speed is not a crucial issue and you'd like to keep memory usage low.

Note that `ZEND_SET_SYMBOL` makes use of the Zend executor globals via the macro `EG`. By specifying `EG(active_symbol_table)`, you get access to the currently active symbol table, dealing with the active, local scope. The local scope may differ depending on whether the function was invoked from within a function.

If you need to optimize for speed and don't care about optimal memory usage, you can omit the check for an existing variable with the same value and instead force insertion into the symbol table by using **zend_hash_update()**:

```
zval *new_variable;

/* allocate and initialize new container */
MAKE_STD_ZVAL(new_variable);

/* set type and variable contents here, see the following sections */

/* introduce this variable by the name "new_variable_name" into the symbol ta-
ble */
zend_hash_update(
```

```
    EG(active_symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

This is actually the standard method used in most modules.

The variables generated with the snippet above will always be of local scope, so they reside in the context in which the function has been called. To create new variables in the global scope, use the same method but refer to another symbol table:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global sym-
bol table
ZEND_SET_SYMBOL(&EG(symbol_table), "new_variable_name", new_variable);
```

The macro `ZEND_SET_SYMBOL` is now being called with a reference to the main, global symbol table by referring `EG(symbol_table)`.

Note: The active_symbol_table variable is a pointer, but symbol_table is not. This is why you have to use `EG(active_symbol_table)` and `&EG(symbol_table)` as parameters to `ZEND_SET_SYMBOL` - it requires a pointer.

Similarly, to get a more efficient version, you can hardcode the symbol table update:

```
zval *new_variable;

// allocate and initialize new container
MAKE_STD_ZVAL(new_variable);

//
// set type and variable contents here
//

// introduce this variable by the name "new_variable_name" into the global sym-
bol table
zend_hash_update(
    &EG(symbol_table),
    "new_variable_name",
    strlen("new_variable_name") + 1,
    &new_variable,
    sizeof(zval *),
    NULL
);
```

Esempio 34-1 shows a sample source that creates two variables - local_variable with a local scope and global_variable with a global scope (see Figure 9.7). The full example can be found on the CD-ROM.

Note: You can see that the global variable is actually not accessible from within the function. This is because it's not imported into the local scope using `global $global_variable;` in the PHP source.

Esempio 34-1. Creating variables with different scopes.

```
ZEND_FUNCTION(variable_creation)
{
    zval *new_var1, *new_var2;

    MAKE_STD_ZVAL(new_var1);
    MAKE_STD_ZVAL(new_var2);

    ZVAL_LONG(new_var1, 10);
    ZVAL_LONG(new_var2, 5);

    ZEND_SET_SYMBOL(EG(active_symbol_table), "local_variable", new_var1);
    ZEND_SET_SYMBOL(&EG(symbol_table), "global_variable", new_var2);

    RETURN_NULL();

}
```

Longs (Integers)

Now let's get to the assignment of data to variables, starting with longs. Longs are PHP's integers and are very simple to store. Looking at the zval.value container structure discussed earlier in this chapter, you can see that the long data type is directly contained in the union, namely in the lval field. The corresponding type value for longs is `IS_LONG` (see Esempio 34-2).

Esempio 34-2. Creation of a long.

```
zval *new_long;

MAKE_STD_ZVAL(new_long);

new_long->type = IS_LONG;
new_long->value.lval = 10;
```

Alternatively, you can use the macro `ZVAL_LONG`:

```
zval *new_long;

MAKE_STD_ZVAL(new_long);
ZVAL_LONG(new_long, 10);
```

Doubles (Floats)

Doubles are PHP's floats and are as easy to assign as longs, because their value is also contained directly in the union. The member in the zval.value container is dval; the corresponding type is `IS_DOUBLE`.

```
zval *new_double;

MAKE_STD_ZVAL(new_double);

new_double->type = IS_DOUBLE;
new_double->value.dval = 3.45;
```

Alternatively, you can use the macro `ZVAL_DOUBLE`:

```
zval *new_double;

MAKE_STD_ZVAL(new_double);
ZVAL_DOUBLE(new_double, 3.45);
```

Strings

Strings need slightly more effort. As mentioned earlier, all strings that will be associated with Zend's internal data structures need to be allocated using Zend's own memory-management functions. Referencing of static strings or strings allocated with standard routines is not allowed. To assign strings, you have to access the structure str in the zval.value container. The corresponding type is `IS_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);

new_string->type = IS_STRING;
new_string->value.str.len = strlen(string_contents);
new_string->value.str.val = estrdup(string_contents);
```

Note the usage of Zend's **estrdup()** here. Of course, you can also use the predefined macro `ZVAL_STRING`:

```
zval *new_string;
char *string_contents = "This is a new string variable";

MAKE_STD_ZVAL(new_string);
ZVAL_STRING(new_string, string_contents, 1);
```

`ZVAL_STRING` accepts a third parameter that indicates whether the supplied string contents should be duplicated (using **estrdup()**). Setting this parameter to `1` causes the string to be duplicated; `0` simply uses the supplied pointer for the variable contents. This is most useful if you want to create a new variable referring to a string that's already allocated in Zend internal memory.

If you want to truncate the string at a certain position or you already know its length, you can use `ZVAL_STRINGL(zval, string, length, duplicate)`, which accepts an explicit string length to be set for the new string. This macro is faster than `ZVAL_STRING` and also binary-safe.

To create empty strings, set the string length to `0` and use `empty_string` as contents:

```
new_string->type = IS_STRING;
new_string->value.str.len = 0;
new_string->value.str.val = empty_string;
```

Of course, there's a macro for this as well (`ZVAL_EMPTY_STRING`):

```
MAKE_STD_ZVAL(new_string);
ZVAL_EMPTY_STRING(new_string);
```

Booleans

Booleans are created just like longs, but have the type `IS_BOOL`. Allowed values in lval are `0` and `1`:

```
zval *new_bool;

MAKE_STD_ZVAL(new_bool);

new_bool->type = IS_BOOL;
new_bool->value.lval = 1;
```

The corresponding macros for this type are `ZVAL_BOOL` (allowing specification of the value) as well as `ZVAL_TRUE` and `ZVAL_FALSE` (which explicitly set the value to `TRUE` and `FALSE`, respectively).

Arrays

Arrays are stored using Zend's internal hash tables, which can be accessed using the **zend_hash_*()** API. For every array that you want to create, you need a new hash table handle, which will be stored in the ht member of the zval.value container.

There's a whole API solely for the creation of arrays, which is extremely handy. To start a new array, you call **array_init()**.

```
zval *new_array;

MAKE_STD_ZVAL(new_array);

if(array_init(new_array) != SUCCESS)
{
    // do error handling here
}
```

If **array_init()** fails to create a new array, it returns `FAILURE`.

To add new elements to the array, you can use numerous functions, depending on what you want to do. Tabella 34-1, Tabella 34-2 and Tabella 34-3 describe these functions. All functions return `FAILURE` on failure and `SUCCESS` on success.

Tabella 34-1. Zend's API for Associative Arrays

Function	Description
add_assoc_long(zval *array, char *key, long n);()	Adds an element of type `long`.
add_assoc_unset(zval *array, char *key);()	Adds an unset element.
add_assoc_bool(zval *array, char *key, int b);()	Adds a Boolean element.
add_assoc_resource(zval *array, char *key, int r);()	Adds a resource to the array.
add_assoc_double(zval *array, char *key, double d);()	Adds a floating-point value.
add_assoc_string(zval *array, char *key, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_assoc_stringl(zval *array, char *key, char *str, uint length, int duplicate); ()	Adds a string with the desired length length to the array. Otherwise, behaves like **add_assoc_string()**.

Tabella 34-2. Zend's API for Indexed Arrays, Part 1

Function	Description
add_index_long(zval *array, uint idx, long n);()	Adds an element of type `long`.
add_index_unset(zval *array, uint idx);()	Adds an unset element.
add_index_bool(zval *array, uint idx, int b);()	Adds a Boolean element.
add_index_resource(zval *array, uint idx, int r);()	Adds a resource to the array.
add_index_double(zval *array, uint idx, double d);()	Adds a floating-point value.

add_index_string(zval *array, uint idx, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_index_stringl(zval *array, uint idx, char *str, uint length, int duplicate);()	Adds a string with the desired length length to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.

Tabella 34-3. Zend's API for Indexed Arrays, Part 2

Function	Description
add_next_index_long(zval *array, long n);()	Adds an element of type `long`.
add_next_index_unset(zval *array);()	Adds an unset element.
add_next_index_bool(zval *array, int b);()	Adds a Boolean element.
add_next_index_resource(zval *array, int r);()	Adds a resource to the array.
add_next_index_double(zval *array, double d);()	Adds a floating-point value.
add_next_index_string(zval *array, char *str, int duplicate);()	Adds a string to the array. The flag duplicate specifies whether the string contents have to be copied to Zend internal memory.
add_next_index_stringl(zval *array, char *str, uint length, int duplicate);()	Adds a string with the desired length length to the array. This function is faster and binary-safe. Otherwise, behaves like **add_index_string()()**.

All these functions provide a handy abstraction to Zend's internal hash API. Of course, you can also use the hash functions directly - for example, if you already have a zval container allocated that you want to insert into an array. This is done using **zend_hash_update()()** for associative arrays (see Esempio 34-3) and **zend_hash_index_update()** for indexed arrays (see Esempio 34-4):

Esempio 34-3. Adding an element to an associative array.

```
zval *new_array, *new_element;
char *key = "element_key";

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

if(array_init(new_array) == FAILURE)
{
    // do error handling here
}

ZVAL_LONG(new_element, 10);

if(zend_hash_update(new_array->value.ht, key, strlen(key) + 1, (void *)&new_element, size
{
    // do error handling here
}
```

Esempio 34-4. Adding an element to an indexed array.

```
zval *new_array, *new_element;
int key = 2;

MAKE_STD_ZVAL(new_array);
MAKE_STD_ZVAL(new_element);

if(array_init(new_array) == FAILURE)
{
    // do error handling here
}

ZVAL_LONG(new_element, 10);

if(zend_hash_index_update(new_array->value.ht, key, (void *)&new_element, sizeof(zval *),
{
    // do error handling here
}
```

To emulate the functionality of **add_next_index_*()**, you can use this:

```
zend_hash_next_index_insert(ht, zval **new_element, sizeof(zval *), NULL)
```

Note: To return arrays from a function, use **array_init()** and all following actions on the predefined variable return_value (given as argument to your exported function; see the earlier discussion of the call interface). You do not have to use `MAKE_STD_ZVAL` on this.

Tip: To avoid having to write `new_array->value.ht` every time, you can use `HASH_OF(new_array)`, which is also recommended for compatibility and style reasons.

Objects

Since objects can be converted to arrays (and vice versa), you might have already guessed that they have a lot of similarities to arrays in PHP. Objects are maintained with the same hash functions, but there's a different API for creating them.

To initialize an object, you use the function **object_init()**:

```
zval *new_object;

MAKE_STD_ZVAL(new_object);

if(object_init(new_object) != SUCCESS)
{
    // do error handling here
}
```

You can use the functions described in Tabella 34-4 to add members to your object.

Tabella 34-4. Zend's API for Object Creation

Function	Description
add_property_long(zval *object, char *key, long l);()	Adds a long to the object.
add_property_unset(zval *object, char *key);()	Adds an unset property to the
add_property_bool(zval *object, char *key, int b);()	Adds a Boolean to the object
add_property_resource(zval *object, char *key, long r);()	Adds a resource to the object
add_property_double(zval *object, char *key, double d);()	Adds a double to the object.
add_property_string(zval *object, char *key, char *str, int duplicate);()	Adds a string to the object.
add_property_stringl(zval *object, char *key, char *str, uint length, int duplicate);()	Adds a string of the specified
add_property_zval(zval *obect, char *key, zval *container):()	Adds a `zval` container to th

Resources

Resources are a special kind of data type in PHP. The term *resources* doesn't really refer to any special kind of data, but to an abstraction method for maintaining any kind of information. Resources are kept in a special resource list within Zend. Each entry in the list has a correspondending type definition that denotes the kind of resource to which it refers. Zend then internally manages all references to this resource. Access to a resource is never possible directly - only via a provided API. As soon as all references to a specific resource are lost, a corresponding shutdown function is called.

For example, resources are used to store database links and file descriptors. The *de facto* standard implementation can be found in the MySQL module, but other modules such as the Oracle module also make use of resources.

> **Nota:** In fact, a resource can be a pointer to anything you need to handle in your functions (e.g. pointer to a structure) and the user only has to pass a single resource variable to your function.

To create a new resource you need to register a resource destruction handler for it. Since you can store any kind of data as a resource, Zend needs to know how to free this resource if its not longer needed. This works by registering your own resource destruction handler to Zend which in turn gets called by Zend whenever your resource can be freed (whether manually or automatically). Registering your resource handler within Zend returns you the **resource type handle** for that resource. This handle is needed whenever you want to access a resource of this type later and is most of time stored in a global static variable within your extension. There is no need to worry about thread safety here because you only register your resource handler once during module initialization.

The Zend function to register your resource handler is defined as:

```
ZEND_API int zend_register_list_destructors_ex(rsrc_dtor_func_t ld, rsrc_dtor_func_t pld,
```

There are two different kinds of resource destruction handlers you can pass to this function: a handler for normal resources and a handler for persistent resources. Persistent resources are for example used for database connection. When registering a resource, either of these handlers must be given. For the other handler just pass `NULL`.

zend_register_list_destructors_ex() accepts the following parameters:

`ld`	Normal resource destruction handler callback
`pld`	Pesistent resource destruction handler callback
`type_name`	A string specifying the name of your resource. It's always a good thing to specify an unique name wi
`module_number`	The `module_number` is automatically available in your `PHP_MINIT_FUNCTION` function and therefo

The return value is an unique integer ID for your **resource type**.

The resource destruction handler (either normal or persistent resources) has the following prototype:

```
void resource_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC);
```

The passed `rsrc` is a pointer to the following structure:

```
typedef struct _zend_rsrc_list_entry {

    void *ptr;
    int type;
    int refcount;

} zend_rsrc_list_entry;
```

The member `void *ptr` is the actual pointer to your resource.

Now we know how to start things, we define our own resource we want register within Zend. It is only a simple structure with two integer members:

```
typedef struct {

    int resource_link;
    int resource_type;

} my_resource;
```

Our resource destruction handler is probably going to look something like this:

```
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    // You most likely cast the void pointer to your structure type

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;

    // Now do whatever needs to be done with you resource. Closing
    // Files, Sockets, freeing additional memory, etc.
    // Also, don't forget to actually free the memory for your resource too!

    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}
```

> **Nota:** One important thing to mention: If your resource is a rather complex structure which also contains pointers to memory you allocated during runtime you have to free them **before** freeing the resource itself!

Now that we have defined

1. what our resource is and
2. our resource destruction handler

we can go on and do the rest of the steps:

1. create a global variable within the extension holding the resource ID so it can be accessed from every function which needs it
2. define the resource name
3. write the resource destruction handler
4. and finally register the handler

```
// Somewhere in your extension, define the variable for your registered resources.
// If you wondered what 'le' stands for: it simply means 'list entry'.
static int le_myresource;

// It's nice to define your resource name somewhere
#define le_myresource_name  "My type of resource"

[...]

// Now actually define our resource destruction handler
void my_destruction_handler(zend_rsrc_list_entry *rsrc TSRMLS_DC) {

    my_resource *my_rsrc = (my_resource *) rsrc->ptr;
    do_whatever_needs_to_be_done_with_the_resource(my_rsrc);
}

[...]

PHP_MINIT_FUNCTION(my_extension) {

    // Note that 'module_number' is already provided through the
    // PHP_MINIT_FUNCTION() function definition.

    le_myresource = zend_register_resource_destructors_ex(my_destruction_handler, NUL

    // You can register additional resources, initialize
    // your global vars, constants, whatever.
}
```

To actually register a new resource you use can either use the **zend_register_resource()** function or the **ZEND_REGISTER_RESOURE()** macro, both defined in zend_list.h . Although the arguments for both map 1:1 it's a good idea to always use macros to be upwards compatible:

```
int ZEND_REGISTER_RESOURCE(zval *rsrc_result, void *rsrc_pointer, int rsrc_type);
```

`rsrc_result`	This is an already initialized `zval *` container.

`rsrc_pointer`	Your resource pointer you want to store.
`rsrc_type`	The type which you received when you registered the resource destruction handler. If you followed the

The return value is an unique integer identifier for that resource.

What is really going on when you register a new resource is it gets inserted in an internal list in Zend and the result is just stored in the given `zval *` container:

```
rsrc_id = zend_list_insert(rsrc_pointer, rsrc_type);

if (rsrc_result) {
    rsrc_result->value.lval = rsrc_id;
    rsrc_result->type = IS_RESOURCE;
}

return rsrc_id;
```

The returned `rsrc_id` uniquly identifies the newly registered resource. You can use the macro `RETURN_RESOURE` to return it to the user:

```
RETURN_RESOURCE(rsrc_id)
```

Nota: It is common practice that if you want to return the resource immidiately to the user you specify the `return_value` as the `zval *` container.

Zend now keeps track of all references to this resource. As soon as all references to the resource are lost, the destructor that you previously registered for this resource is called. The nice thing about this setup is that you don't have to worry about memory leakages introduced by allocations in your module - just register all memory allocations that your calling script will refer to as resources. As soon as the script decides it doesn't need them anymore, Zend will find out and tell you.

Now that the user got his resource, at some point he is passing it back to one of your functions. The value.lval inside the `zval *` container contains the key to your resource and thus can be used to fetch the resource with the following macro: `ZEND_FETCH_RESOURCE`:

```
ZEND_FETCH_RESOURCE(rsrc, rsrc_type, rsrc_id, default_rsrc_id, resource_type_name, resou
```

`rsrc`	This is your pointer which will point to your previously registered resource.
`rsrc_type`	This is the typecast argument for your pointer, e.g. `myresource *`.
`rsrc_id`	This is the address of the `zval *`container the user passed to your function, e.g. `&z_resourc`
`default_rsrc_id`	This integer specifies the default resource `ID` if no resource could be fetched or -1.
`resource_type_name`	This is the name of the requested resource. It's a string and is used when the resource can't be
`resource_type`	The `resource_type` you got back when registering the resource destruction handler. In our e

This macro has no return value. It is for the developers convenience and takes care of TSRMLS arguments passing and also does check if the resource could be fetched. It throws a warning message and returns the current PHP function with `NULL` if there was a problem retrieving the resource.

To force removal of a resource from the list, use the function **zend_list_delete()**. You can also force

the reference count to increase if you know that you're creating another reference for a previously allocated value (for example, if you're automatically reusing a default database link). For this case, use the function **zend_list_addref()**. To search for previously allocated resource entries, use **zend_list_find()**. The complete API can be found in `zend_list.h`.

Macros for Automatic Global Variable Creation

In addition to the macros discussed earlier, a few macros allow easy creation of simple global variables. These are nice to know in case you want to introduce global flags, for example. This is somewhat bad practice, but Table Tabella 34-5 describes macros that do exactly this task. They don't need any zval allocation; you simply have to supply a variable name and value.

Tabella 34-5. Macros for Global Variable Creation

Macro	Description
`SET_VAR_STRING(name, value)`	Creates a new string.
`SET_VAR_STRINGL(name, value, length)`	Creates a new string of the specified length. This macro is faster than `SET_VAR_STRING` and also binary-safe.
`SET_VAR_LONG(name, value)`	Creates a new long.
`SET_VAR_DOUBLE(name, value)`	Creates a new double.

Creating Constants

Zend supports the creation of true constants (as opposed to regular variables). Constants are accessed without the typical dollar sign (`$`) prefix and are available in all scopes. Examples include `TRUE` and `FALSE`, to name just two.

To create your own constants, you can use the macros in Tabella 34-6. All the macros create a constant with the specified name and value.

You can also specify flags for each constant:

- `CONST_CS` - This constant's name is to be treated as case sensitive.
- `CONST_PERSISTENT` - This constant is persistent and won't be "forgotten" when the current process carrying this constant shuts down.

To use the flags, combine them using a inary OR:

```
// register a new constant of type "long"
    REGISTER_LONG_CONSTANT("NEW_MEANINGFUL_CONSTANT", 324, CONST_CS |
    CONST_PERSISTENT);
```

There are two types of macros - `REGISTER_*_CONSTANT` and`REGISTER_MAIN_*_CONSTANT`. The first type creates constants bound to the current module. These constants are dumped from the symbol table as soon as the module that registered the constant is unloaded from memory. The second type creates constants that remain in the symbol table independently of the module.

Tabella 34-6. Macros for Creating Constants

Macro	
`REGISTER_LONG_CONSTANT(name, value, flags)`	`REGISTER_MAIN_LONG_CONSTANT(name, value, flags`
`REGISTER_DOUBLE_CONSTANT(name, value, flags)`	`REGISTER_MAIN_DOUBLE_CONSTANT(name, value, f`
`REGISTER_STRING_CONSTANT(name, value, flags)`	`REGISTER_MAIN_STRING_CONSTANT(name, value, f`
`REGISTER_STRINGL_CONSTANT(name, value, length, flags)`	`REGISTER_MAIN_STRINGL_CONSTANT(name`

Capitolo 35. Duplicating Variable Contents: The Copy Constructor

Sooner or later, you may need to assign the contents of one zval container to another. This is easier said than done, since the zval container doesn't contain only type information, but also references to places in Zend's internal data. For example, depending on their size, arrays and objects may be nested with lots of hash table entries. By assigning one zval to another, you avoid duplicating the hash table entries, using only a reference to them (at most).

To copy this complex kind of data, use the *copy constructor*. Copy constructors are typically defined in languages that support operator overloading, with the express purpose of copying complex types. If you define an object in such a language, you have the possibility of overloading the "=" operator, which is usually responsible for assigning the contents of the lvalue (result of the evaluation of the left side of the operator) to the rvalue (same for the right side).

Overloading means assigning a different meaning to this operator, and is usually used to assign a function call to an operator. Whenever this operator would be used on such an object in a program, this function would be called with the lvalue and rvalue as parameters. Equipped with that information, it can perform the operation it intends the "=" operator to have (usually an extended form of copying).

This same form of "extended copying" is also necessary for PHP's zval containers. Again, in the case of an array, this extended copying would imply re-creation of all hash table entries relating to this array. For strings, proper memory allocation would have to be assured, and so on.

Zend ships with such a function, called **zend_copy_ctor()** (the previous PHP equivalent was **pval_copy_constructor()**).

A most useful demonstration is a function that accepts a complex type as argument, modifies it, and then returns the argument:

```
zval *parameter;

if (zend_parse_parameters(ZEND_NUM_ARGS() TSRMLS_CC, "z", &parameter) == FAILURE)
   return;
}

// do modifications to the parameter here

// now we want to return the modified container:
*return_value == *parameter;
zval_copy_ctor(return_value);
```

The first part of the function is plain-vanilla argument retrieval. After the (left out) modifications, however, it gets interesting: The container of parameter is assigned to the (predefined) return_value container. Now, in order to effectively duplicate its contents, the copy constructor is called. The copy constructor works directly with the supplied argument, and the standard return values are `FAILURE` on failure and `SUCCESS` on success.

If you omit the call to the copy constructor in this example, both parameter and return_value would point to the same internal data, meaning that return_value would be an illegal additional reference to the same data structures. Whenever changes occurred in the data that parameter points to, return_value might be affected. Thus, in order to create separate copies, the copy constructor must be used.

The copy constructor's counterpart in the Zend API, the destructor **zval_dtor()**, does the opposite of the constructor.

Capitolo 36. Returning Values

Returning values from your functions to PHP was described briefly in an earlier section; this section gives the details. Return values are passed via the return_value variable, which is passed to your functions as argument. The return_value argument consists of a zval container (see the earlier discussion of the call interface) that you can freely modify. The container itself is already allocated, so you don't have to run `MAKE_STD_ZVAL` on it. Instead, you can access its members directly.

To make returning values from functions easier and to prevent hassles with accessing the internal structures of the zval container, a set of predefined macros is available (as usual). These macros automatically set the correspondent type and value, as described in Tabella 36-1 and Tabella 36-2.

> **Nota:** The macros in Tabella 36-1 automatically *return* from your function, those in Tabella 36-2 only *set* the return value; they don't return from your function.

Tabella 36-1. Predefined Macros for Returning Values from a Function

Macro	Description
`RETURN_RESOURCE(resource)`	Returns a resource.
`RETURN_BOOL(bool)`	Returns a Boolean.
`RETURN_NULL()`	Returns nothing (a NULL value).
`RETURN_LONG(long)`	Returns a long.
`RETURN_DOUBLE(double)`	Returns a double.
`RETURN_STRING(string, duplicate)`	Returns a string. The duplicate flag indicates whether the string should be duplicated using **estrdup()**.
`RETURN_STRINGL(string, length, duplicate)`	Returns a string of the specified length; otherwise, behaves like `RETURN_STRING`. This macro is faster and binary-safe, however.
`RETURN_EMPTY_STRING()`	Returns an empty string.
`RETURN_FALSE`	Returns Boolean false.
`RETURN_TRUE`	Returns Boolean true.

Tabella 36-2. Predefined Macros for Setting the Return Value of a Function

Macro	Description
`RETVAL_RESOURCE(resource)`	Sets the return value to the specified resource.
`RETVAL_BOOL(bool)`	Sets the return value to the specified Boolean value.
`RETVAL_NULL`	Sets the return value to NULL.
`RETVAL_LONG(long)`	Sets the return value to the specified long.
`RETVAL_DOUBLE(double)`	Sets the return value to the specified double.
`RETVAL_STRING(string, duplicate)`	Sets the return value to the specified string and duplicates it to Zend internal memory if desired (see also `RETURN_STRING`).

`RETVAL_STRINGL(string, length, duplicate)`	Sets the return value to the specified string and forces the length to become length (see also `RETVAL_STRING`). This macro is faster and binary-safe, and should be used whenever the string length is known.
`RETVAL_EMPTY_STRING`	Sets the return value to an empty string.
`RETVAL_FALSE`	Sets the return value to Boolean false.
`RETVAL_TRUE`	Sets the return value to Boolean true.

Complex types such as arrays and objects can be returned by using **array_init()** and **object_init()**, as well as the corresponding hash functions on return_value. Since these types cannot be constructed of trivial information, there are no predefined macros for them.

Capitolo 37. Printing Information

Often it's necessary to print messages to the output stream from your module, just as print() would be used within a script. PHP offers functions for most generic tasks, such as printing warning messages, generating output for phpinfo(), and so on. The following sections provide more details. Examples of these functions can be found on the CD-ROM.

zend_printf()

zend_printf() works like the standard printf(), except that it prints to Zend's output stream.

zend_error()

zend_error() can be used to generate error messages. This function accepts two arguments; the first is the error type (see `zend_errors.h`), and the second is the error message.

```
zend_error(E_WARNING, "This function has been called with empty arguments");
```

Tabella 37-1 shows a list of possible values (see Figura 37-1). These values are also referred to in `php.ini`. Depending on which error type you choose, your messages will be logged.

Tabella 37-1. Zend's Predefined Error Messages.

Error	Description
`E_ERROR`	Signals an error and terminates execution of the script immediately .
`E_WARNING`	Signals a generic warning. Execution continues.
`E_PARSE`	Signals a parser error. Execution continues.
`E_NOTICE`	Signals a notice. Execution continues. Note that by default the display of this type of error mes
`E_CORE_ERROR`	Internal error by the core; shouldn't be used by user-written modules.
`E_COMPILE_ERROR`	Internal error by the compiler; shouldn't be used by user-written modules.
`E_COMPILE_WARNING`	Internal warning by the compiler; shouldn't be used by user-written modules.

Figura 37-1. Display of warning messages in the browser.

Including Output in phpinfo()

After creating a real module, you'll want to show information about the module in phpinfo() (in addition to the module name, which appears in the module list by default). PHP allows you to create your own section in the phpinfo() output with the `ZEND_MINFO()` function. This function should be placed in the module descriptor block (discussed earlier) and is always called whenever a script calls phpinfo().

PHP automatically prints a section in phpinfo() for you if you specify the `ZEND_MINFO` function, including the module name in the heading. Everything else must be formatted and printed by you.

Typically, you can print an HTML table header using **php_info_print_table_start()** and then use the standard functions **php_info_print_table_header()** and **php_info_print_table_row()**. As arguments, both take the number of columns (as integers) and the column contents (as strings). Esempio 37-1 shows a source example and its output. To print the table footer, use **php_info_print_table_end()**.

Esempio 37-1. Source code and screenshot for output in phpinfo().

```
php_info_print_table_start();
php_info_print_table_header(2, "First column", "Second column");
php_info_print_table_row(2, "Entry in first row", "Another entry");
php_info_print_table_row(2, "Just to fill", "another row here");
php_info_print_table_end();
```

Execution Information

You can also print execution information, such as the current file being executed. The name of the function currently being executed can be retrieved using the function **get_active_function_name()**. This function returns a pointer to the function name and doesn't accept any arguments. To retrieve the name of the file currently being executed, use **zend_get_executed_filename()**. This function accesses the executor globals, which are passed to it using the `TSRMLS_C` macro. The executor globals are automatically available to every function that's called directly by Zend (they're part of the `INTERNAL_FUNCTION_PARAMETERS` described earlier in this chapter). If you want to access the executor globals in another function that doesn't have them available automatically, call the macro `TSRMLS_FETCH()` once in that function; this will introduce them to your local scope.

Finally, the line number currently being executed can be retrieved using the function **zend_get_executed_lineno()**. This function also requires the executor globals as arguments. For examples of these functions, see Esempio 37-2.

Esempio 37-2. Printing execution information.

```
zend_printf("The name of the current function is %s<br>", get_active_function_name(TSRMLS
zend_printf("The file currently executed is %s<br>", zend_get_executed_filename(TSRMLS_C)
zend_printf("The current line being executed is %i<br>", zend_get_executed_lineno(TSRMLS_
```

Capitolo 38. Startup and Shutdown Functions

Startup and shutdown functions can be used for one-time initialization and deinitialization of your modules. As discussed earlier in this chapter (see the description of the Zend module descriptor block), there are global, module, and request startup and shutdown events.

The global startup functions are called once when PHP starts up; similarly, the global shutdown functions are called once when PHP shuts down. Please note that they're really only called *once*, not when a new Apache process is being created!

The module startup and shutdown functions are called whenever a module is loaded and needs initialization; the request startup and shutdown functions are called every time a request is processed (meaning that a file is being executed).

For dynamic extensions, module and request startup/shutdown events happen at the same time.

Declaration and implementation of these functions can be done with macros; see the earlier section "Declaration of the Zend Module Block" for details.

Capitolo 39. Calling User Functions

You can call user functions from your own modules, which is very handy when implementing callbacks; for example, for array walking, searching, or simply for event-based programs.

User functions can be called with the function **call_user_function_ex()**. It requires a hash value for the function table you want to access, a pointer to an object (if you want to call a method), the function name, return value, number of arguments, argument array, and a flag indicating whether you want to perform zval separation.

```
ZEND_API int call_user_function_ex(HashTable *function_table, zval *object,
zval *function_name, zval **retval_ptr_ptr,
int param_count, zval **params[],
int no_separation);
```

Note that you don't have to specify both function_table and object; either will do. If you want to call a method, you have to supply the object that contains this method, in which case **call_user_function()**automatically sets the function table to this object's function table. Otherwise, you only need to specify function_table and can set object to `NULL`.

Usually, the default function table is the "root" function table containing all function entries. This function table is part of the compiler globals and can be accessed using the macro `CG`. To introduce the compiler globals to your function, call the macro `TSRMLS_FETCH` once.

The function name is specified in a zval container. This might be a bit surprising at first, but is quite a logical step, since most of the time you'll accept function names as parameters from calling functions within your script, which in turn are contained in zval containers again. Thus, you only have to pass your arguments through to this function. This zval must be of type `IS_STRING`.

The next argument consists of a pointer to the return value. You don't have to allocate memory for this container; the function will do so by itself. However, you have to destroy this container (using **zval_dtor()**) afterward!

Next is the parameter count as integer and an array containing all necessary parameters. The last argument specifies whether the function should perform zval separation - this should always be set to `0`. If set to `1`, the function consumes less memory but fails if any of the parameters need separation.

Esempio 39-1 shows a small demonstration of calling a user function. The code calls a function that's supplied to it as argument and directly passes this function's return value through as its own return value. Note the use of the constructor and destructor calls at the end - it might not be necessary to do it this way here (since they should be separate values, the assignment might be safe), but this is bulletproof.

Esempio 39-1. Calling user functions.

```
zval **function_name;
zval *retval;

if((ZEND_NUM_ARGS() != 1) || (zend_get_parameters_ex(1, &function_name) != SUCCESS))
{
    WRONG_PARAM_COUNT;
}

if((*function_name)->type != IS_STRING)
{
    zend_error(E_ERROR, "Function requires string argument");
}

TSRMSLS_FETCH();
```

```
if(call_user_function_ex(CG(function_table), NULL, *function_name, &retval, 0, NULL, 0)
{
    zend_error(E_ERROR, "Function call failed");
}

zend_printf("We have %i as type<br>", retval->type);

*return_value = *retval;
zval_copy_ctor(return_value);
zval_ptr_dtor(&retval);
```

```
<?php

dl("call_userland.so");

function test_function()
{

    print("We are in the test function!<br>");

    return("hello");

}

$return_value = call_userland("test_function");

print("Return value: \"$return_value\"<br>");
?>
```

Capitolo 40. Initialization File Support

PHP 4 features a redesigned initialization file support. It’s now possible to specify default initialization entries directly in your code, read and change these values at runtime, and create message handlers for change notifications.

To create an .ini section in your own module, use the macros `PHP_INI_BEGIN()` to mark the beginning of such a section and `PHP_INI_END()` to mark its end. In between you can use `PHP_INI_ENTRY()` to create entries.

```
PHP_INI_BEGIN()
PHP_INI_ENTRY("first_ini_entry",  "has_string_value", PHP_INI_ALL, NULL)
PHP_INI_ENTRY("second_ini_entry", "2",                PHP_INI_SYSTEM, OnChangeSecond)
PHP_INI_ENTRY("third_ini_entry",  "xyz",              PHP_INI_USER, NULL)
PHP_INI_END()
```

The `PHP_INI_ENTRY()` macro accepts four parameters: the entry name, the entry value, its change permissions, and a pointer to a change-notification handler. Both entry name and value must be specified as strings, regardless of whether they really are strings or integers.

The permissions are grouped into three sections:`PHP_INI_SYSTEM` allows a change only directly in the `php3.ini` file; `PHP_INI_USER` allows a change to be overridden by a user at runtime using additional configuration files, such as `.htaccess`; and `PHP_INI_ALL` allows changes to be made without restrictions. There’s also a fourth level, `PHP_INI_PERDIR`, for which we couldn’t verify its behavior yet.

The fourth parameter consists of a pointer to a change-notification handler. Whenever one of these initialization entries is changed, this handler is called. Such a handler can be declared using the `PHP_INI_MH` macro:

```
PHP_INI_MH(OnChangeSecond);             // handler for ini-entry "second_ini_entry"

// specify ini-entries here

PHP_INI_MH(OnChangeSecond)
{

    zend_printf("Message caught, our ini entry has been changed to %s<br>", new_value);

    return(SUCCESS);

}
```

The new value is given to the change handler as string in the variable new_value. When looking at the definition of `PHP_INI_MH`, you actually have a few parameters to use:

```
#define PHP_INI_MH(name) int name(php_ini_entry *entry, char *new_value,
                                  uint new_value_length, void *mh_arg1,
                                  void *mh_arg2, void *mh_arg3)
```

All these definitions can be found in `php_ini.h`. Your message handler will have access to a structure that contains the full entry, the new value, its length, and three optional arguments. These optional arguments can be specified with the additional macros `PHP_INI_ENTRY1` (allowing one additional argument), `PHP_INI_ENTRY2` (allowing two additional arguments), and `PHP_INI_ENTRY3` (allowing three additional arguments).

The change-notification handlers should be used to cache initialization entries locally for faster access or to perform certain tasks that are required if a value changes. For example, if a constant

connection to a certain host is required by a module and someone changes the hostname, automatically terminate the old connection and attempt a new one.

Access to initialization entries can also be handled with the macros shown in Tabella 40-1.

Tabella 40-1. Macros to Access Initialization Entries in PHP

Macro	Description
`INI_INT(name)`	Returns the current value of entry `name` as integer (long).
`INI_FLT(name)`	Returns the current value of entry `name` as float (double).
`INI_STR(name)`	Returns the current value of entry `name` as string. *Note:* This string is not duplicated, but inst
`INI_BOOL(name)`	Returns the current value of entry `name` as Boolean (defined as zend_bool, which currently m
`INI_ORIG_INT(name)`	Returns the original value of entry `name` as integer (long).
`INI_ORIG_FLT(name)`	Returns the original value of entry `name` as float (double).
`INI_ORIG_STR(name)`	Returns the original value of entry `name` as string. Note: This string is not duplicated, but ins
`INI_ORIG_BOOL(name)`	Returns the original value of entry `name` as Boolean (defined as zend_bool, which currently m

Finally, you have to introduce your initialization entries to PHP. This can be done in the module startup and shutdown functions, using the macros `REGISTER_INI_ENTRIES()` and `UNREGISTER_INI_ENTRIES()`:

```
ZEND_MINIT_FUNCTION(mymodule)
{

    REGISTER_INI_ENTRIES();

}

ZEND_MSHUTDOWN_FUNCTION(mymodule)
{

    UNREGISTER_INI_ENTRIES();

}
```

Capitolo 41. Where to Go from Here

You've learned a lot about PHP. You now know how to create dynamic loadable modules and statically linked extensions. You've learned how PHP and Zend deal with internal storage of variables and how you can create and access these variables. You know quite a set of tool functions that do a lot of routine tasks such as printing informational texts, automatically introducing variables to the symbol table, and so on.

Even though this chapter often had a mostly "referential" character, we hope that it gave you insight on how to start writing your own extensions. For the sake of space, we had to leave out a lot; we suggest that you take the time to study the header files and some modules (especially the ones in the `ext/standard` directory and the MySQL module, as these implement commonly known functionality). This will give you an idea of how other people have used the API functions - particularly those that didn't make it into this chapter.

Capitolo 42. Reference: Some Configuration Macros

config.m4

The file `config.m4` is processed by `buildconf` and must contain all the instructions to be executed during configuration. For example, these can include tests for required external files, such as header files, libraries, and so on. PHP defines a set of macros that can be used in this process, the most useful of which are described in Tabella 42-1.

Tabella 42-1. M4 Macros for `config.m4`

Macro	Description
`AC_MSG_CHECKING(message)`	Prints a "checking <mess
`AC_MSG_RESULT(value)`	Gives the result to `AC_MS`
`AC_MSG_ERROR(message)`	Prints message as error m
`AC_DEFINE(name,value,description)`	Adds `#define` to `php_c`
`AC_ADD_INCLUDE(path)`	Adds a compiler include
`AC_ADD_LIBRARY_WITH_PATH(libraryname,librarypath)`	Specifies an additional lib
`AC_ARG_WITH(modulename,description,unconditionaltest,conditionaltest)`	Quite a powerful macro, a
`PHP_EXTENSION(modulename, [shared])`	This macro is a *must* to ca

Capitolo 43. API Macros

A set of macros was introduced into Zend's API that simplify access to zval containers (see Tabella 43-1).

Tabella 43-1. API Macros for Accessing zval Containers

Macro	Refers to
Z_LVAL(zval)	(zval).value.lval
Z_DVAL(zval)	(zval).value.dval
Z_STRVAL(zval)	(zval).value.str.val
Z_STRLEN(zval)	(zval).value.str.len
Z_ARRVAL(zval)	(zval).value.ht
Z_LVAL_P(zval)	(*zval).value.lval
Z_DVAL_P(zval)	(*zval).value.dval
Z_STRVAL_P(zval_p)	(*zval).value.str.val
Z_STRLEN_P(zval_p)	(*zval).value.str.len
Z_ARRVAL_P(zval_p)	(*zval).value.ht
Z_LVAL_PP(zval_pp)	(**zval).value.lval
Z_DVAL_PP(zval_pp)	(**zval).value.dval
Z_STRVAL_PP(zval_pp)	(**zval).value.str.val
Z_STRLEN_PP(zval_pp)	(**zval).value.str.len
Z_ARRVAL_PP(zval_pp)	(**zval).value.ht

Parte VI. FAQ: Frequently Asked Questions (domande e risposte ricorrenti)

Capitolo 44. Informazioni Generali

Questa sezione contiene le domande più generali riguardanti il PHP: cos'è e cosa fa.

1. Cos'è il PHP?

Dal manuale (http://www.php.net/manual/):

PHP è un linguaggio di script immerso nel HTML. Molta della sua sintassi è presa in prestito dai linguaggi C, Java e Perl, a cui sono state aggiunte alcune specifiche caratteristiche del PHP. L'obiettivo del linguaggio è di semplificare il lavoro dei webmaster nella realizzazione di pagine dinamiche.

Una buona introduzione al PHP, a cura di Stig Sæther Bakken, può essere trovata qui (http://www.zend.com/zend/art/intro.php) sul sito web di Zend. Inoltre, gran parte del materiale usato per le conferenze sul PHP (http://conf.php.net/) è liberamente disponibile.

2. Cosa significa PHP?

PHP significa *PHP: Hypertext Preprocessor*. Questo confonde molte persone poich[la prima parola dell'acronimo è l'acronimo stesso. Questo tipo di acronimo è chiamato acronimo ricorsivo. Chi è curioso può visitare Free On-Line Dictionary of Computing (http://www.foldoc.org/) per avere maggiori informazioni riguardanti gli acronimi ricorsivi.

3. Qual'è la relazione fra le varie versioni?

PHP/FI 2.0 è una vecchia versione di PHP non più supportata. PHP 3 è il successore di PHP/FI 2.0 ed è molto più gradevole. PHP 4 è l'ultima generazione di PHP, che al suo interno fa uso del motore Zend (http://www.zend.com/).

4. Si possono avere in esecuzione contemporaneamente diverse versioni di PHP?

Sì. Fare riferimento al file `INSTALL` incluso nella distribuzione del codice sorgente di PHP 4. Fare inoltre riferimento alla relativa appendice.

5. Quali sono le differenze fra PHP 3 e PHP 4?

Sono disponibili un paio di articoli (http://www.zend.com/zend/art/) scritti dagli autori stessi di PHP 4. Questa è una lista di alcune delle più importanti nuove caratteristiche:

- Modulo API esteso
- Sotto UNIX, processo di compilazione generalizzato
- Interfaccia generica verso i web server, anche verso quelli che supportano il multi-threading
- Migliore evidenziatore di sintassi
- Supporto nativo per le sessioni HTTP
- Supporto per il buffering dell'output
- Sistema di configurazione più potente
- Reference counting

Per favore fare riferimento a panoramica delle novità in PHP 4 (http://www.zend.com/zend/whats-new.php) per un'esposizione dettagliata di queste nuove caratteristiche e altro ancora. Se si è in procinto di migrare dal PHP 3 al PHP 4 si consiglia di leggere la relativa appendice.

6. Credo di avere trovato un bug! A chi lo devo dire?

Dovresti andare a visitare il database dei Bug del PHP e assicurarti che il bug non sia già conosciuto. Se non lo vedi nel database, usa il modulo per inviare il bug. È importante usare il database dei bug invece di mandare una email ad una delle mailing list, perché in questo caso al bug viene assegnato un tracking number e sarà per te possibile tornare più tardi a verificare lo stato del bug. Il database dei bug può essere trovato qui http://bugs.php.net/.

Capitolo 45. Mailing list

Questa sezione contiene domande su come fare per entrare in contatto con la comunità PHP. Il modo migliore è tramite le mailing list.

1. Esistono mailing list dedicate al PHP?

Certo! Ci sono molte mailing list dedicate a svariati aspetti. Una lista completa di mailing list può essere trovata nella nostra pagina dedicata al Supporto (http://www.php.net/support.php).

La mailing list più generica è `php-general`. Per iscriversi, inviare una mail a php-general-subscribe@lists.php.net (mailto:php-general-subscribe@lists.php.net). Non è necessario scrivere nulla come oggetto o come corpo del messaggio. Per cancellarsi, inviare una mail a php-general-unsubscribe@lists.php.net (mailto:php-general-unsubscribe@lists.php.net).

Ci si può iscrivere e cancellare usando l'interfaccia web nella nostra pagina di Supporto (http://www.php.net/support.php) .

2. Ci sono altre comunità?

Ve ne sono numerosissime in giro per il mondo. Abbiamo link di alcuni server IRC e di mailing list in lingua straniera nella nostra pagina di Support (http://www.php.net/support.php).

3. Aiuto! Non risco a iscrivermi/cancellarmi a/da una delle mailing list!

Se si riscontrano problemi nell'iscriversi o nel cancellarsi dalla mailing list php-general, può essere perché il software che gestisce la mailing list stessa non riesce a individuare l'indirizzo corretto da usare. Se l'indirizzo email è `pluto@pippo.com`, si può mandare una richiesta di iscrizione a `php-general-subscribe-pluto=pippo.com@lists.php.net`, o la richiesta di cancellazione a `php-general-unsubscribe-pluto=pippo.com@lists.php.net`. Utilizzare indirizzi corrispondenti per le altre mailing list.

4. Esiste un archivio delle mailing list?

Sì, una lista dei siti che ospitano gli archivi si trova nella pagina di Supporto (http://www.php.net/support.php). Gli articoli delle mailinglist vengono anche archiviati come messaggi news. Si può accedere al news server news://news.php.net/ usando un client per le news. Esiste anche un'interfaccia web sperimentale verso il news server http://news.php.net/

5. Cosa si può chiedere nelle mailing list?

Poiché PHP cresce di popolarità di giorno in giorno, il traffico sulla mailing list php-general è notevolmente aumentato, attualmente la lista riceve circa 150-200 messaggi al giorno. Considerato ciò, è nell'interesse di tutti usare la lista solo come ultima chance, cioè solo quando si è già cercato in tutti gli altri posti.

Prima di inviare un messaggio alla lista, bisognerebbe dare una lettura a queste FAQ e al manuale per vedere se si può trovare aiuto lì. Se non si trova nulla, si può provare con gli archivi della mailing list (vedere sopra). Se si riscontrano problemi nell'installare o nel configurare PHP, per favore fare riferimento alla documentazione inclusa nella distribuzione e ai vari README. Se anche allora non si riescono a trovare informazioni sufficienti, si è i benvenuti ad usare la mailing list.

6. Quali informazioni si dovrebbero includere nel mandare un messaggio a una mailinglist?

Messaggi tipo: "Non riesco a installare PHP! Aiuto! Cosa c'è di sbagliato?" non sono di aiuto a nessuno. Se si hanno problemi a installare PHP, si dovrebbero includere il sistema operativo nel quale lo si sta cercando di installare, quale versione di PHP si vuole installare, in quale forma lo si ha

a disposizione (pre-compilato, dal CVS, RPM e così via), cosa si è tentato fino ad ora, dove ci si è bloccati e l'esatto messaggio di errore.

Questa regola è applicabile agli altri problemi che si possino incontrare. Si devono includere informazioni su cosa si è fatto, dove si è rimasti bloccati, cosa si sta cercando di fare e , se possibile, fornire gli esatti messaggi di errore. Se avete problemi con il vostro codice sorgente, dovreste includere la parte di codice che non è funzionante. Non includere più codice di quello che è strettamente necessario! Quello renderebbe il messaggio difficile da leggere e molte persone lo salterebbero proprio a causa di questo. Se siete poco sicuri su quante informazioni includere nel messaggio, meglio includerne di più che di meno.

Un'altra cosa importante da ricordare è quella di sintetizzare il vostro problema nell'oggetto del messaggio. Un oggetto del tipo "AIUTATEMIIIIIIIII!!!" oppure "Dove sta problema?" verrà ignorato dalla maggioranza dei lettori.